

The figure consists of a central 'YYY' symbol surrounded by a cross of 'SSS' symbols. This central cross is enclosed within two concentric circles of 'SSSSSSSS' symbols. The entire figure is oriented vertically, with the top of the figure pointing downwards.

SSSSSSSS YY YY SSSSSSSS LL NN NN MM MM
SSSSSSSS YY YY SSSSSSSS LL NN NN MM MM
SS YY YY SS SS LL NN NN MMMM MMMM
SS YY YY SS SS LL NN NN MMMM MMMM
SS YY YY SS SS LL NNNN NN MM MM MM
SS YY YY SSSSSS LL NN NN MM MM
SS YY SSSSSS LL NN NN MM MM
SS YY SS LL NN NNNN MM MM
SS YY SS LL NN NNNN MM MM
SS YY SS LL NN NN MM MM
SS YY SSSSSS LL LLLLLLLL NN NN MM MM
SSSSSSSS YY SSSSSSSS LLLLLLLL NN NN MM MM
SSSSSSSS YY SSSSSSSS LLLLLLLL NN NN MM MM
.....

LL IIIIII SSSSSSSS
LL IIIIII SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II IIIIII SSSSSSSS
LL II IIIIII SSSSSSSS

(2)	511	DECLARATIONS
(3)	620	EXE\$CRELNNT - CREATE LOGICAL NAME TABLE
(4)	1206	EXE\$CRELNMM - CREATE LOGICAL NAME
(5)	1819	EXE\$DELLNM - DELETE LOGICAL NAME
(6)	2070	EXESTRNLNM - TRANSLATE LOGICAL NAME

0000 1 .TITLE SYSLNM - SYSTEM SERVICES TO MANIPULATE LOGICAL NAMES AND TABLES.
0000 2 .IDENT 'V04-000'
0000 3 :
0000 4 :
0000 5 :*****
0000 6 :
0000 7 : COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 : DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 : ALL RIGHTS RESERVED.
0000 10 :
0000 11 : THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 : ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 : INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 : COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 : OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 : TRANSFERRED.
0000 17 :
0000 18 : THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 : AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 : CORPORATION.
0000 21 :
0000 22 : DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 : SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :
0000 25 :
0000 26 :*****
0000 27 :
0000 28 : TODD M. KATZ 01-APR-83
0000 29 :
0000 30 : SYSTEM SERVICES TO MANIPULATE LOGICAL NAMES
0000 31 :
0000 32 : CREATE LOGICAL NAME TABLES
0000 33 : CREATE LOGICAL NAME
0000 34 : DELETE LOGICAL NAME
0000 35 : TRANSLATE LOGICAL NAME
0000 36 :
0000 37 : MODIFICATION HISTORY:
0000 38 :
0000 39 : V03-027 RAS0327 Ron Schaefer 24-Jul-1984
0000 40 : Back out RAS0322 and make the LNMS_LENGTH itemcode
0000 41 : be a longword value.
0000 42 :
0000 43 : V03-026 RAS0322 Ron Schaefer 10-Jul-1984
0000 44 : Fix \$TRNLNM item LNMS_LENGTH to only write a word field.
0000 45 :
0000 46 : V03-025 RAS0312 Ron Schaefer 21-Jun-1984
0000 47 : Fix ORB alignment within logical name table.
0000 48 :
0000 49 : V03-024 TMK0014 Todd M. Katz 21-Apr-1984
0000 50 : The interface to the internal logical name routine
0000 51 : LNMSDELETE_LNMB has been changed. Update \$DELLNM, which calls
0000 52 : this routine, to reflect this new interface.
0000 53 :
0000 54 : The performance measurement cell used to monitor the rate of
0000 55 : logical name translations is currently located within the
0000 56 : internal logical name routine LNMSSEARCHLOG. Unfortunately,
0000 57 : because of its current placement, any attempts to delete

0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :

specific logical names will also increment this counter. This is because the system service SDELLNM will call the routine LNMSSEARCHLOG in such a situation. Therefore, in order to be able to make a more accurate measurement of the overall rate of logical name translations, I have decided to move this performance measurement cell from its current single location to several more appropriate locations. One of these new locations is within the system service STRNLNM, just before the call to LNMSSEARCHLOG.

V03-023 TMK0013 Todd M. Katz 29-Mar-1984
Modify the logical name system services to make use of the updated internal protection checking mechanisms. What this requires is a modification to SYSSCRELNT so that all shareable logical name tables are created with a quad-word aligned Object Rights Block in place of a un-aligned CHIP protection template.

Restrict all names which appear with a directory table to 31 characters consisting of the DEC multi-national alphanumeric character set plus \$ and -. This restriction applies to both logical names and logical name table names. The reason for this restriction is that in the future we might want to support hierachial name spaces. If we decide to do so we would have to invent and then impose a structure on logical name table names. This restriction gives us sufficient leeway (and more importantly sufficient available characters) to be able to define a hierachial name space structure in the future.

V03-022 TMK0012 Todd M. Katz 07-Mar-1984
The fixed portion of each and every translation block has been increased by a word, LNMXSW_HASH, in order to potentially hold the translation's hash code value. Furthermore, SYSSCRELNT and SYSSCRELNM have been modified so that this field within each and every translation block is initialized to 0. This hash code field will be used in an optimization of logical name table name processing. As such, only certain logical names, those that are contained within the process or system directory and may be used in logical name table processing, need to have the hash code fields of each of their translations initialized to their equivalence string's hash code value. This initialization actually takes place within the routine LNMSINSLOGTAB in the module LNMSUB.

V03-021 TMK0011 Todd M. Katz 13-Feb-1984
Add SSS_NOLOGTAB to the list of errors than can be returned by SYSSCRELNT, SYSSCRELNM, and SYSSDELLNM.

Fix a bug in STRNLNM. At the present time the very first longword in the item list is not being probed. Obviously this could result in a disasterous kernel mode ACCVIO.

Fix up STRNLNM's processing of the PARENT item. The most important change I have made here, besides a general fixup of the code, is to return 0 bytes in the parent item if the LNMB being looked at is not for a logical name table. This is because logical names, with the exception of logical name tables, do not have parents.

0000 115
0000 116
0000 117
0000 118
0000 119
0000 120
0000 121
0000 122
0000 123
0000 124
0000 125
0000 126
0000 127
0000 128
0000 129
0000 130
0000 131
0000 132
0000 133
0000 134
0000 135
0000 136
0000 137
0000 138
0000 139
0000 140
0000 141
0000 142
0000 143
0000 144
0000 145
0000 146
0000 147
0000 148
0000 149
0000 150
0000 151
0000 152
0000 153
0000 154
0000 155
0000 156
0000 157
0000 158
0000 159
0000 160
0000 161
0000 162
0000 163
0000 164
0000 165
0000 166
0000 167
0000 168
0000 169
0000 170
0000 171

Optimize the processing of \$TRNLNM's item list. This is done by having two separate code segments - one to probe and return variable length character string items (TABLE, STRING, PARENT), and the other to probe and return longword length items (LENGTH, ATTRIBUTES, MAX_INDEX). In addition, since ACMODEs are only a byte, ACMODE items are now being probed and filled in in-line. Also, optimize a few instructions within this system service.

V03-020 TMK0010 Todd M. Katz 29-Dec-1983
Re-write SYSSDELLNM for the case when the name of the logical name table entry to be deleted has been specified. What was being done was incorrect. It allowed the possibility of deleting several logical names from different tables. The correct way to implement this is as follows:

1. Search for the first instance of a logical name table entry in one of the specified logical name tables which possesses the specified access mode.
2. Check to make sure that the caller has write access to the containing logical name table.
3. Delete the logical name table entry and all outermode aliases within the same logical name table.

The process and system directory logical name tables are now being created with the LNMB\$V NODELETE bit set within their LNMB\$B FLAGS fields to indicate that these tables should never be deleted. Therefore, it is no longer necessary for SYSSCRELNT to set this bit within the appropriate directory table to protect it from being deleted during the insertion of the table entry for the new logical name table, and conversely SYSSCRELNT must never clear this bit within a directory table after the attempt at insertion has been made.

Fix a bug in the creation of logical names without translations by SYSSCRELNM. This used to work, but has broken mysteriously. The problem is that during the first processing of the item list, the number of translations specified by the list and the cumulated size of their translation strings is saved on the stack for use in computing the amount of storage required to be allocated for the new logical name block. These stack associated counters were not being appropriately initialized to zero in the case when no item list (and thus no translations) was present. This resulted in the size of the table name and the address of the table name buffer being popped off the stack and used as the values of these counters. As might be expected this would cause the system service to fail in a variety of interesting ways. Often, since the saved logical name table name descriptor had been removed from the stack but the pointer to the saved descriptor in R9 had not changed, this would lead to a failure to find a logical name table and termination of the system service. Also, since the values used in computing the amount of storage required for the new logical name were random, the value computed could be so large as to cause the storage allocation attempt to fail.

0000 172
0000 173
0000 174
0000 175
0000 176
0000 177
0000 178
0000 179
0000 180
0000 181
0000 182
0000 183
0000 184
0000 185
0000 186
0000 187
0000 188
0000 189
0000 190
0000 191
0000 192
0000 193
0000 194
0000 195
0000 196
0000 197
0000 198
0000 199
0000 200
0000 201
0000 202
0000 203
0000 204
0000 205
0000 206
0000 207
0000 208
0000 209
0000 210
0000 211
0000 212
0000 213
0000 214
0000 215
0000 216
0000 217
0000 218
0000 219
0000 220
0000 221
0000 222
0000 223
0000 224
0000 225
0000 226
0000 227
0000 228

V03-019 TMK0009 Todd M. Katz 18-Dec-1983
Remove all mention of LNMSV_SYSTEM and LNMSV_GROUP from SYSSCRELNT. The system table and all group tables will be handcrafted, and there will not be a way to create such a table by means of the system service interface.
Also, change some PUSHRs into PUSHLs (or MOVQs) and POPRs into POPLs (or MOVQs) where appropriate for performance reasons.

V03-018 TMK0008 Todd M. Katz 26-Oct-1983
Quota checking when logical name tables are being created is presently incorrect. Currently, when a logical name table is created only one type of quota check is made. That check makes sure that the quota holder of the parent logical name table has sufficient quota for both the logical name table itself and any quota which will be specifically allocated to it. Actually, two quota checks should be made. The quota holder of the parent logical name table should have sufficient quota for the quota that will be specifically allocated to the new table; but in addition, the containing table (the system or process directory table) must have sufficient quota for the logical name table itself. This is consistent with how quota deductions are made for logical names, and it is consistent with the philosophy that logical name tables are just logical names with a special type of translation.

V03-017 TMK0007 Todd M. Katz 26-Sep-1983
Change the default protection that is assigned to new shareable logical name tables to SYSTEM:RWED OWNER:RWED GROUP: WORLD: so that by default, the system can access and modify any shareable logical name table.

V03-016 TMK0006 Todd M. Katz 16-Sep-1983
Fix a branch in EXESCRELNT. If no attributes were specified, this service should branch to check out the table name parameter; however, it is branching into the wrong place. This results in an inability to create a logical name table if attributes are not specified. This fixes the problem.

V03-015 TMK0005 Todd M. Katz 08-Aug-1983
Make several modifications to each of the logical name system services.

Changes to EXESCRELNT:

1. Change access mode processing. The new logical name table may not be created in an access mode inner to that of the caller unless the user has SYSNAM privilege and has specified an access mode as an optional system service argument. A new logical name table may still be created with an access mode outer to the mode of the caller, even if the caller does not have SYSNAM privilege, if the caller explicitly specifies an access mode as an optional system service parameter.

2. A new logical name table maybe marked within the table

0000 229 :
0000 230 :
0000 231 :
0000 232 :
0000 233 :
0000 234 :
0000 235 :
0000 236 :
0000 237 :
0000 238 :
0000 239 :
0000 240 :
0000 241 :
0000 242 :
0000 243 :
0000 244 :
0000 245 :
0000 246 :
0000 247 :
0000 248 :
0000 249 :
0000 250 :
0000 251 :
0000 252 :
0000 253 :
0000 254 :
0000 255 :
0000 256 :
0000 257 :
0000 258 :
0000 259 :
0000 260 :
0000 261 :
0000 262 :
0000 263 :
0000 264 :
0000 265 :
0000 266 :
0000 267 :
0000 268 :
0000 269 :
0000 270 :
0000 271 :
0000 272 :
0000 273 :
0000 274 :
0000 275 :
0000 276 :
0000 277 :
0000 278 :
0000 279 :
0000 280 :
0000 281 :
0000 282 :
0000 283 :
0000 284 :
0000 285 :

header as either the system table or a group table (but not both) by setting one of the new attribute bits (reserved to DIGITAL) LNMSV_SYSTEM or LNMSV_GROUP respectively. The caller must be in executive or kernel mode to do so.

3. It is necessary to have quota access to the parent logical name table in order to be able create a subtable. In addition if the table being creating is shareable, and the caller is specifying a name, then the caller must have write access to the system directory table. Previously, read and write access to the parent logical name table was what was being checked. This is not only insufficient but wrong. I have also moved these access checks so that they are performed immediately after the parent logical name table is located.
4. SYSNAM was formerly required to specify the name of a shareable logical name table and not let the system default name it (ie - insert the name into the system directory table). The meaning of SYSNAM has been restricted to refer only to the access of the system logical name table, LNMSYSTEM_TABLE, and to the creation and deletion of inner access mode logical names and tables.
5. If no protection mask was specified, then default the protection to 0:RWED.
6. It is never necessary to do any protection checking when process-private logical name tables are involved. Therefore, do not allocate or fill in the CHIP protection structure template when creating a process-private logical name table, and do not perform any protection checking during the creation of such tables.

Changes to EXE\$CRELNM:

1. Change access mode processing. The new logical name may not be created in an access mode inner to that of the caller unless the user has SYSNAM privilege and has specified an access mode as an optional system service argument. A new logical name may still be created with an access mode outer to the mode of the caller, even if the caller does not have SYSNAM privilege, if the caller explicitly specifies an access mode as an optional system service parameter.
2. It is only necessary to have write access to the logical name table in order to be able to insert a logical name within it. Previously, both read and write access was required.
3. In order to be able to insert a logical name into the system directory table a caller must be able to write access the table itself. Previously, the caller could perform such an insertion if they had SYSNAM privilege. The meaning of SYSNAM has been restricted to refer only to the access of the system logical name table, LNMSYSTEM_TABLE, and to the creation and deletion of inner access mode logical names and

0000 286 :
0000 287 :
0000 288 :
0000 289 :
0000 290 :
0000 291 :
0000 292 :
0000 293 :
0000 294 :
0000 295 :
0000 296 :
0000 297 :
0000 298 :
0000 299 :
0000 300 :
0000 301 :
0000 302 :
0000 303 :
0000 304 :
0000 305 :
0000 306 :
0000 307 :
0000 308 :
0000 309 :
0000 310 :
0000 311 :
0000 312 :
0000 313 :
0000 314 :
0000 315 :
0000 316 :
0000 317 :
0000 318 :
0000 319 :
0000 320 :
0000 321 :
0000 322 :
0000 323 :
0000 324 :
0000 325 :
0000 326 :
0000 327 :
0000 328 :
0000 329 :
0000 330 :
0000 331 :
0000 332 :
0000 333 :
0000 334 :
0000 335 :
0000 336 :
0000 337 :
0000 338 :
0000 339 :
0000 340 :
0000 341 :
0000 342 :

tables.

4. It is never necessary to do any protection checking when process-private logical name tables are involved. Therefore, do not perform any protection checking prior to the insertion of new logical names into such tables.

Changes to EXE\$DELLNM:

1. Change access mode processing. Logical names or logical name tables in access modes inner to that of the caller may not be deleted unless the user has SYSNAM privilege and has specified an access mode as an optional system service argument. Logical names or logical name tables in access modes outer to the mode of the caller may still be deleted if the caller explicitly specifies an access mode as an optional system service parameter, even if the caller does not have SYSNAM privilege.
2. When a single logical name table entry was to be deleted this system service was requiring delete access to the containing logical name table before allowing the deletion to proceed. When all logical name table entries within a logical name table were to be deleted, no access checking was being done. This is incorrect. The access requirements for the deletion of logical name tables and logical names are as follows:
 - a. To delete a logical name, write access to the containing logical name table is required.
 - b. To delete a logical name table, either delete access to the table itself or write access to its containing logical name table (the system directory table) is required.
3. It is never necessary to do any protection checking when process-private logical name tables are involved. Therefore, do not perform any protection checking prior to the deletion of any process-private logical name table entry.

Changes to EXE\$TRNLNM:

1. This system service would accept an invalid access mode as an optional system service parameter without returning an error. Add access mode error checking.
2. It is never necessary to do any protection checking when process-private logical name tables are involved. Therefore, do not perform any protection checking if the target logical name table entry is process-private.

V03-014 RAS0166 Ron Schaefer 5-Jul-1983
Reverse polarity of branch in positioning to a translation index within a STRNLNM itmlst. Also optimize #-1s.

V03-013 WMC0001 Wayne Cardoza 23-Jun-1983
Add chained item lists, parent item code.

0000 343 :
0000 344 :
0000 345 :
0000 346 :
0000 347 :
0000 348 :
0000 349 :
0000 350 :
0000 351 :
0000 352 :
0000 353 :
0000 354 :
0000 355 :
0000 356 :
0000 357 :
0000 358 :
0000 359 :
0000 360 :
0000 361 :
0000 362 :
0000 363 :
0000 364 :
0000 365 :
0000 366 :
0000 367 :
0000 368 :
0000 369 :
0000 370 :
0000 371 :
0000 372 :
0000 373 :
0000 374 :
0000 375 :
0000 376 :
0000 377 :
0000 378 :
0000 379 :
0000 380 :
0000 381 :
0000 382 :
0000 383 :
0000 384 :
0000 385 :
0000 386 :
0000 387 :
0000 388 :
0000 389 :
0000 390 :
0000 391 :
0000 392 :
0000 393 :
0000 394 :
0000 395 :
0000 396 :
0000 397 :
0000 398 :
0000 399 :

V03-012 RAS0160 Ron Schaefer 16-Jun-1983
Charge all arguments to be by-reference. This affects
ACMODE all services
ATTR \$CRELNM, \$CRELNNT and \$STRNLNM
PROMSK \$CRELNNT
QUOTA \$CRELNNT

V03-011 DMW4046 DMWalg 9-Jun-1983
Post intergration of new logical name structures
1.) MAX_INDEX in TRNLNM returns the maximum index
if non-negative, else -1
2.) Clean up the setting default index value of
zero in TRNLNM
3.) Replace ^xFFFF with LNMS_LNMB_ADDR as item value

V03-010 DMW4038 DMWalg 25-May-1983
Fix check for LNMS_LNMB_ADDR item

V03-009 RAS0158 Ron Schaefer 25-May-1983
Add protection structure and checking to logical
name tables. Currently, only SOGW protection is implemented.
Correct item list buffer probing in \$TRNLNM to make
it tolerate longer than necessary buffers.

V03-008 DMW4029 DMWalg 25-May-1983
Added code to CRELNFM to allow MTL and mailbox UCB to be
intergrated with the new logical name structures. This
includes the addition of LNMS_LNMB_ADDR and work to allow
LNMS_C_BACKPTR to be accepted thru system service interface.

V03-007 TMK0004 Todd M. Katz 30-Apr-1983
When VMS must default name a new logical name table, it is
given a name of the form LNMSxxxxxxxx instead of a name of
the form LNTxxxxxxxx.

V03-006 TMK0003 Todd M. Katz 25-Apr-1983
Make multiple to changes to the routines within this module:
EXE\$CRELNNT:
1. Un-comment the SETIPL that was commented out for debugging
purposes.
2. Change several instructions to conform to VMS coding
requirements.
3. If a process-private logical name table is to be created,
the size of the logical name block is saved on the stack,
the system pool logical name block is deallocated, the size
is retrieved, and process-private P1 space is allocated. I
was saving this size as a word on the stack, and this was
creating problems allocating P1 space, after the size was
popped off the stack because I neglected to zero out the
high order word. To fix this problem, size is now saved as
a longword on the stack.

0000 400 :
0000 401 :
0000 402 :
0000 403 :
0000 404 :
0000 405 :
0000 406 :
0000 407 :
0000 408 :
0000 409 :
0000 410 :
0000 411 :
0000 412 :
0000 413 :
0000 414 :
0000 415 :
0000 416 :
0000 417 :
0000 418 :
0000 419 :
0000 420 :
0000 421 :
0000 422 :
0000 423 :
0000 424 :
0000 425 :
0000 426 :
0000 427 :
0000 428 :
0000 429 :
0000 430 :
0000 431 :
0000 432 :
0000 433 :
0000 434 :
0000 435 :
0000 436 :
0000 437 :
0000 438 :
0000 439 :
0000 440 :
0000 441 :
0000 442 :
0000 443 :
0000 444 :
0000 445 :
0000 446 :
0000 447 :
0000 448 :
0000 449 :
0000 450 :
0000 451 :
0000 452 :
0000 453 :
0000 454 :
0000 455 :
0000 456 :

4. R2 is now saved before calling LNMSLOCKW (which destroys it) instead of after.
5. Instead of picking up the QUOTA argument twice from the user's parameter list, it is picked up and saved the first time it is needed.
6. Pickup ACMODE as a byte instead of a longword.
7. Change the name of TABNAM to RESNAM, TABLEN to RESLEN, and LOGNAM to TABNAM.
8. If the new table is mapped to an existing table, then save the address of the existing LNMB on the stack before deleting the new LNMB so that the table's name maybe returned if required.

EXESCRELNM:

1. Fix an incorrect probing of the TABNAM parameter.
2. Un-comment the SETIPL that was commented out for debugging purposes.
3. Do not perform any validation of the item list (except for the STRING item) during the first pass.
4. Return SSS_NOGRPNAME instead of SSS_NOPRIV if the user does not have sufficient privilege to create a shareable logical name.
5. If a process-private logical name is to be created, the size of the logical name block is saved on the stack, the system pool logical name block is deallocated, the size is retrieved, and process-private P1 space is allocated. I was saving this size as a word on the stack, and this was creating problems allocating P1 space, after the size was popped off the stack because I neglected to zero out the high order word. To fix this problem, size is now saved as a longword on the stack.
6. Pickup the address of all item buffers before probing. After probing refer only to system service copies of these addresses, and not to user copies.
7. As each translation block is created, check to make sure that creation of this block would not result in the exceeding of the space allocated for the logical name block.
8. Change several instructions to conform to VMS coding requirements.
9. Pickup ACMODE as a byte instead of a longword.

EXESDELLNM:

1. Un-comment the SETIPL that was commented out for debugging

0000 457 :
0000 458
0000 459
0000 460
0000 461
0000 462
0000 463
0000 464
0000 465
0000 466
0000 467
0000 468
0000 469
0000 470
0000 471
0000 472
0000 473
0000 474
0000 475
0000 476
0000 477
0000 478
0000 479
0000 480
0000 481
0000 482
0000 483
0000 484
0000 485
0000 486
0000 487
0000 488
0000 489
0000 490
0000 491
0000 492
0000 493
0000 494
0000 495
0000 496
0000 497
0000 498
0000 499
0000 500
0000 501
0000 502
0000 503
0000 504
0000 505
0000 506
0000 507
0000 508
0000 509 --

purposes.

2. When the deletion of a specific logical name in an access mode other than user has been requested, all outer access mode logical names with the same name and in the same table are also deleted.

3. Pickup ACMODE as a byte instead of a longword.

EXE\$TRNLNM:

1. Fix an incorrect probing of the TABNAM parameter.
2. Un-comment the SETIPL that was commented out for debugging purposes.
3. Pickup the address of all item buffers before probing. After probing refer only to system service copies of these addresses, and not to user copies.
4. Change how the index of INDEX items are validated.
5. Change several instructions to conform to VMS coding requirements.
6. Return the attribute bit LNMSV_SHAREABLE in the ATTRIBUTES item whenever the name being translated is shareable.
7. Pickup ACMODE as a byte instead of a longword.
8. Change how the ACMODE argument is interpreted. If the argument is 0, maximize with the mode of the caller; otherwise, the specified mode is used to qualify the search.
9. Change the attribute bit LNMSV NOT EXIST to LNMSV EXISTS. This bit will be set in the ATTRIBUTES item buffers of those indexes that do exist, and clear in the ATTRIBUTES item buffers of those indexes that do not exist.

V03-005 BLS0219 Benn Schreiber 19-Apr-1983
Make some BSBW's into JSB's with longword displacements.

V03-004 TMK0002 Todd M. Katz 18-Apr-1983
Fix a broken ASSUME statement. This ASSUME broke when I changed the structure of a table header by removing the field LNMTHSL LOGNAM. Also, no longer fill in this "non-existent" field when creating a logical name table.

V03-003 TMK0001 Todd M. Katz 18-Mar-1983
Re-write SYSSCRELNT. Add SYSSCRELN, SYSSDELLNM, SYS\$TRNLNM.

0000 511 .SBTTL DECLARATIONS
0000 512
0000 513 : MACRO LIBRARY CALLS
0000 514
0000 515 :
0000 516
0000 517 \$ARMDEF :DEFINE ACCESS RIGHTS MASK
0000 518 \$CADEF :DEFINE CONDITIONAL ASSEMBLY SWITCHES
0000 519 \$DYNDEF :DEFINE DATA STRUCTURE TYPE CODES
0000 520 \$IPLDEF :DEFINE INTERRUPT PRIORITY LEVELS
0000 521 \$LNMDDEF :DEFINE LOGICAL NAME ATTRIBUTES
0000 522 \$LNMSRDEF :DEFINE LOGICAL NAME STRUCTURES OFFSETS
0000 523 \$ORBDEF :DEFINE OBJECT RIGHTS BLOCK OFFSETS
0000 524 \$PCBDEF :DEFINE PCB OFFSETS
0000 525 \$PRDEF :DEFINE PROCESSOR REGISTER NUMBERS
0000 526 \$PRVDEF :DEFINE PRIVILEGE BITS
0000 527 \$PSLDEF :DEFINE PROCESSOR STATUS FIELDS
0000 528 \$SSDEF :DEFINE SYSTEM STATUS VALUES
0000 529
0000 530 : LOCAL SYMBOLS
0000 531
0000 532 : ARGUMENT LIST OFFSET DEFINITIONS FOR CREATE LOGICAL NAME TABLE.
0000 533
0000 534 :
0000 535
00000004 536 CTATTR=4 :ADDRESS OF TABLE ATTRIBUTES
00000008 537 CTRESNAM=8 :ADDRESS OF TABLE NAME STRING DESCRIPTOR
0000000C 538 CTRESLEN=12 :ADDRESS OF WORD TO RECEIVE LENGTH OF TABLE NAME
00000010 539 CTQUOTA=16 :ADDRESS OF TABLE QUOTA
00000014 540 CTPROT=20 :ADDRESS OF PROTECTION MASK
00000018 541 CTTABNAM=24 :ADDRESS OF TABLE NAME STRING DESCRIPTOR
0000001C 542 CTPARTAB=28 :PARENT TABLE NAME DESCRIPTOR
00000020 543 CTACMODE=32 :ADDRESS OF ACCESS MODE
0000 544
0000 545 : ARGUMENT LIST OFFSET DEFINITIONS FOR CREATE LOGICAL NAME.
0000 546
0000 547 :
0000 548
00000004 549 CNATTR=4 :ADDRESS OF ATTRIBUTES
00000008 550 CNTABNAM=8 :ADDRESS OF TABLE NAME STRING DESCRIPTOR
0000000C 551 CNLOGNAM=12 :ADDRESS OF LOGICAL NAME STRING DESCRIPTOR
00000010 552 CNACMODE=16 :ADDRESS OF ACCESS MODE
00000014 553 CNITMLST=20 :ADDRESS OF ITEM LIST
0000 554
0000 555 : ARGUMENT LIST OFFSET DEFINITIONS FOR DELETE LOGICAL NAME.
0000 556
0000 557 :
0000 558
00000004 559 DNTABNAM=4 :ADDRESS OF TABLE NAME STRING DESCRIPTOR
00000008 560 DNLOGNAM=8 :ADDRESS OF LOGICAL NAME STRING DESCRIPTOR
0000000C 561 DNACMODE=12 :ADDRESS OF ACCESS MODE
0000 562
0000 563 :
0000 564 : ARGUMENT LIST OFFSET DEFINITIONS FOR TRANSLATE LOGICAL NAME.
0000 565 :
0000 566 :
00000004 567 TRATTR=4 :ADDRESS OF TRANSLATION FLAGS

00000008 0000 568 TRTABNAM=8 ;ADDRESS OF TABLE NAME STRING DESCRIPTOR
0000000C 0000 569 TRLOGNAM=12 ;ADDRESS OF LOGICAL NAME STRING DESCRIPTOR
00000010 0000 570 TRACMODE=16 ;ADDRESS OF ACCESS MODE
00000014 0000 571 TRITMLST=20 ;ADDRESS OF ITEM LIST
00000000 572
00000000 573 : PROTECTION MASK CONSTANTS FOR CHECKING.
00000000 574 :
00000000 575 :
00000001 0000 577 READ_ACCESS = ARMSM_READ
00000002 0000 578 WRITE_ACCESS = ARMSM_WRITE
00000004 0000 579 QUOTA_ACCESS = ARMSM_EXECUTE
00000008 0000 580 DELETE_ACCESS = ARMSM_DELETE
00000000 581
00000000 582 : ASSUME STATEMENTS FOR LOGICAL NAME BLOCK, LOGICAL NAME TRANSLATION, AND
00000000 583 : LOGICAL NAME TABLE HEADER.
00000000 584 :
00000000 585 :
00000000 586
00000000 587 ASSUME LNMB\$B_TYPE+1, EQ, LNMB\$B_ACMODE
00000000 588 ASSUME LNMB\$B_ACMODE+1, EQ, LNMB\$L_TABLE
00000000 589 ASSUME LNMB\$L_TABLE+4, EQ, LNMB\$B_FLAGS
00000000 590 ASSUME LNMB\$B_FLAGS+1, EQ, LNMB\$T_NAME
00000000 591
00000000 592 ASSUME LNMX\$B_FLAGS, EQ, 0
00000000 593 ASSUME LNMX\$B_FLAGS+1, EQ, LNMX\$B_INDEX
00000000 594 ASSUME LNMX\$B_INDEX+1, EQ, LNMX\$W_HASH
00000000 595 ASSUME LNMX\$W_HASH+2, EQ, LNMX\$T_XLATION
00000000 596
00000000 597 ASSUME LNMT\$B_FLAGS, EQ, 0
00000000 598 ASSUME LNMT\$B_FLAGS+1, EQ, LNMT\$L_HASH
00000000 599 ASSUME LNMT\$L_HASH+4, EQ, LNMT\$L_ORB
00000000 600 ASSUME LNMT\$L_ORB+4, EQ, LNMT\$L_NAME
00000000 601 ASSUME LNMT\$L_NAME+4, EQ, LNMT\$L_PARENT
00000000 602 ASSUME LNMT\$L_PARENT+4, EQ, LNMT\$L_CHILD
00000000 603 ASSUME LNMT\$L_CHILD+4, EQ, LNMT\$L_SIBLING
00000000 604 ASSUME LNMT\$L_SIBLING+4, EQ, LNMT\$L_QTABLE
00000000 605 ASSUME LNMT\$L_QTABLE+4, EQ, LNMT\$L_BYTESLM
00000000 606 ASSUME LNMT\$L_BYTESLM+4, EQ, LNMT\$L_BYTES
00000000 607
00000000 608 .PSECT YF\$SLNM
00000000 609
00000000 610 : LOCAL DATA
00000000 611 :
00000000 612 :
00000000 613 :
00000000 614 LNM_HEX_TAB_LEN = 4 + 8 + 8 ;SIZE OF "LNMS" + pid + addr
00000000 615 HEXDIGITS:
42 41 39 38 37 36 35 34 33 32 31 30 46 45 44 43 000C 616 .ASCII /0123456789ABCDEF/ ;TABLE TO CONVERT INTEGER TO HEX DIGIT
0010 617
0010 618 : .PAGE

0010 620 .SBTTL EXESCRELNT - CREATE LOGICAL NAME TABLE
0010 621
0010 622 +
0010 623 EXESCRELNT - CREATE LOGICAL NAME TABLE
0010 624
0010 625 THIS SERVICE PROVIDES THE CAPABILITY TO CREATE A LOGICAL NAME TABLE.
0010 626
0010 627 INPUTS:
0010 628
0010 629 CTATTR(AP) = ADDRESS OF TABLE ATTRIBUTES.
0010 630 CTRESNAM(AP) = ADDRESS OF DESCRIPTOR TO RECEIVE TABLE NAME STRING.
0010 631 CTRESLEN(AP) = ADDRESS OF WORD TO RECEIVE LENGTH OF TABLE NAME.
0010 632 CTQUOTA(AP) = ADDRESS OF BYTE QUOTA FOR TABLE AND NAMES CONTAINED THEREIN.
0010 633 CTPROT(AP) = ADDRESS OF SOGW PROTECTION MASK.
0010 634 CTTABNAM(AP) = ADDRESS OF TABLE NAME STRING DESCRIPTOR.
0010 635 CTPARTAB(AP) = ADDRESS OF PARENT TABLE NAME DESCRIPTOR.
0010 636 CTACMODE(AP) = ADDRESS OF ACCESS MODE OF LOGICAL NAME TABLE TO BE CREATED.
0010 637
0010 638 R4 = CURRENT PROCESS PCB ADDRESS.
0010 639
0010 640 OUTPUTS:
0010 641
0010 642 RO LOW BIT CLEAR INDICATES FAILURE TO CREATE LOGICAL NAME TABLE ENTRY.
0010 643
0010 644 RO = SSS_ACCVIO - TABLE NAME DESCRIPTOR, LOGICAL NAME
0010 645 DESCRIPTOR, LOGICAL NAME STRING, PARENT TABLE NAME
0010 646 DESCRIPTOR, PARENT TABLE NAME STRING CANNOT BE READ BY
0010 647 CALLING ACCESS MODE. TABLE NAME LENGTH WORD, TABLE NAME
0010 648 STRING BUFFER CANNOT BE WRITTEN BY CALLING ACCESS MODE.
0010 649
0010 650 RO = SSS_BADPARAM - INVALID ATTRIBUTE OR ACCESS MODE SPECIFIED.
0010 651 PARENT LOGICAL NAME TABLE NAME NOT SPECIFIED.
0010 652
0010 653 RO = SSS_DUPLNAM - ATTEMPT MADE TO SUPERSEDE NON-ALIASABLE
0010 654 LOGICAL NAME TABLE ENTRY OF THE SAME OR AN INNER ACCESS
0010 655 MODE.
0010 656
0010 657 RO = SSS_EXLNMQOUTA - INSUFFICIENT QUOTA AVAILABLE IN THE
0010 658 PARENT LOGICAL NAME TABLE'S QUOTA TABLE FOR THE
0010 659 CREATION OF THE NEW LOGICAL NAME TABLE ENTRY.
0010 660
0010 661 RO = SSS_INSFMEM - SUFFICIENT SYSTEM DYNAMIC MEMORY DOES NOT
0010 662 EXIST TO ALLOCATE THE NEW LOGICAL NAME TABLE ENTRY AND
0010 663 IMPLICIT RESOURCE WAIT IS NOT ENABLED.
0010 664
0010 665 RO = SSS_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
0010 666 LOGICAL OR PARENT TABLE NAME STRING SPECIFIED.
0010 667
0010 668 RO = SSS_IVLOGTAB - INVALID PARENT TABLE NAME SPECIFIED OR
0010 669 INVALID LOGICAL NAME TABLE NAME SPECIFIED.
0010 670
0010 671 RO = SSS_NOLOGTAB - PARENT TABLE SPECIFIED DOES NOT EXIST.
0010 672
0010 673 RO = SSS_NOPRIV - PROCESS DOES NOT HAVE PRIVILEGE TO CREATE
0010 674 SPECIFIED LOGICAL NAME TABLE ENTRY.
0010 675
0010 676 RO = SSS_PARENT_DEL - INSERTION OF NEW LOGICAL NAME TABLE ENTRY.

0010 677 : WOULD HAVE RESULTED IN THE DELETION OF A (GRAND)PARENT
0010 678 : LOGICAL NAME TABLE ENTRY.

0010 679 :
0010 680 : RO = SSS_RESULTOVF - BUFFER NOT LARGE ENOUGH TO CONTAIN NAME
0010 681 : OF NEW LOGICAL NAME TABLE.

0010 682 :
0010 683 : RO = SSS_TOOMANYLNM - TOO MANY LEVELS OF RECURSION IN SEARCH
0010 684 : FOR PARENT TABLE.

0010 685 :
0010 686 : RO LOW BIT SET INDICATES SUCCESSFUL COMPLETION.

0010 687 :
0010 688 : RO = SSS_LNMCREATED - NORMAL COMPLETION, NEW LOGICAL NAME TABLE
0010 689 : ENTRY CREATED.

0010 690 :
0010 691 : RO = SSS_NORMAL - NORMAL COMPLETION, LNM\$V CREATE IF SET AND
0010 692 : THE LOGICAL NAME TABLE ENTRY ALREADY EXISTED.

0010 693 :
0010 694 : RO = SSS_SUPERSEDE - NORMAL COMPLETION, NEW LOGICAL NAME TABLE
0010 695 : ENTRY SUPERSEDED A PREVIOUSLY EXISTING ENTRY AT THE SAME
0010 696 : OR OUTER ACCESS MODE IN THE SPECIFIED PARENT LOGICAL
0010 697 : NAME TABLE.

0010 698 :
0010 699 : SIDE EFFECTS:

0010 700 :
0010 701 : THIS ROUTINE EXITS AT IPL 2, AND MUST CONTINUE TO DO SO, BECAUSE
0010 702 : IT IS CALLED AT SYSTEM INITIALIZATION TIME.

0010 703 :
0010 704 :-
0010 705 :

001F' 0FFC 0000 31 0002 0005	00000000 706 .PSECT YSEXEPAGED
	00000000 707 .ENTRY EXE\$CRELNT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
	00000000 708 BRW EXECRELNT

50 0C 3C 0010 04 0013	0010 710 .PSECT YF\$SLNM
	0010 711 .ENABLE LSB
	0010 712 900\$: MOVZWL #SSS_ACCVIO,RO ;ACCESS VIOLATION
50 14 3C 0014 04 0017	0010 713 RET
	0010 714 910\$: MOVZWL #SSS_BADPARAM,RO ;BAD SYSTEM SERVICE PARAMETER
50 0124 8F 3C 0018 04 001D	0010 715 RET
	0010 716 920\$: MOVZWL #SSS_INSFMEM,RO ;NO PRIVILEGE TO CREATE TABLE ENTRY
50 015C 8F 3C 001E 04 0023	0010 717 930\$: RET
	0010 718 940\$: MOVZWL #SSS_IVLOGTAB,RO ;INVALID LOGICAL NAME TABLE NAME
	0010 719 RET
	0010 720
	0010 721

0024 722 : VALIDATE AND COPY PARAMETERS AS NECESSARY. REGISTER ASSIGNMENT FOR THE
0024 723 : PARAMETERS ARE AS FOLLOWS:

0024 724	R5 = ACCESS MODE.
	0024 725 R6 = PROTECTION MASK.
	0024 726 R7 = ATTRIBUTE BITS.
0024 727	R8 = ADDRESS OF PROBED AND COPIED PARENT TABLE NAME DESCRIPTOR.
	0024 728 R9 = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
0024 729	0024 730 R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
	0024 731 R11 = ADDRESS OF WORD TO RECEIVE TABLE NAME LENGTH.
0024 732	
	0024 733

5B 0C AC D0 0024 734 EXECRELNT:
 06 13 0024 735 MOVL CTRESLEN(AP),R11 ;ADDRESS OF TABLE NAME LENGTH WORD
 0028 736 BEQL 10\$;BRANCH IF NOT PRESENT
 002A 737 IFNOWRT #2,(R11),900\$;CHECK WRITE ACCESS TO WORD

56 14 AC D0 0030 738 10\$: MOVL CTPROT(AP),R6 ;ADDRESS OF PROTECTION MASK WORD
 07 12 0034 740 BNEQ 15\$;BRANCH IF ONE PRESENT
 56 FF00 8F 3C 0036 741 MOVZWL #^XFF00,R6 ;DEFAULT THE PROTECTION TO S:RWED O:RWED
 09 11 003B 742 BRB 20\$
 56 66 3C 0043 743 15\$: IFNORD #2,(R6),900\$;CHECK READ ACCESS TO WORD
 0046 744 MOVZWL (R6),R6 ;GET VALUE

59 08 AC D0 0046 745 20\$: MOVL CTRESNAM(AP),R9 ;ADDRESS OF TABLE NAME DESCRIPTOR
 12 13 004A 747 BEQL 30\$;BRANCH IF NOT PRESENT
 51 59 004C 748 MOVL R9,R1 ;SET UP CALL
 00000000'EF 16 004F 749 JSB L^EXESPROBEW_DSC ;PROBE DESCRIPTOR AND WRITE PROBE BUFFER
 C5 50 E9 0055 750 BLBC R0,930\$;CAN'T WRITE BUFFER OR READ DESCRIPTOR
 7E 51 7D 0058 751 MOVQ R1,-(SP)
 59 5E D0 005B 752 MOVL SP,R9 ;SAVE TABLE NAME DESCRIPTOR
 005E 753 30\$: MOVL CTATTR(AP),R7 ;ADDRESS OF TABLE NAME DESCRIPTOR

57 04 AC D0 005E 754 BEQL 35\$;TABLE ATTRIBUTES
 12 13 0062 755 IFNORD #4,(R7),900\$;BRANCH IF NOT PRESENT
 57 67 0064 756 MOVL (R7),R7 ;CHECK READ ACCESS TO VALUE
 FFFFFFFC 8F D3 006D 757 BITL #^C<- ;GET VALUE

0074 759 LNMSM_CONFINE! - ;CONFINE TO PROCESS
 0074 760 LNMSM_CREATE IF! - ;SUPERCEDE VS. MAP
 0074 761 LNMSM_NO_ALIAS - ;DO NOT ALLOW ALIASES

9E 12 0074 762 >R7
 0076 763 BNEQ 910\$;INVALID TABLE ATTRIBUTES
 0076 764
 0076 765 ASSUME LNMSC_TABNAMLEN,LE,512
 0076 766 35\$: MOVL CTTABNAM(AP),R10 ;ADDRESS OF TABLE NAME DESCRIPTOR
 5A 18 AC D0 0076 767 BEQL 40\$;BRANCH IF NOT PRESENT
 1F 13 007A 768 IFNORD #8,(R10),900\$;CHECK ACCESS TO DESCRIPTOR
 51 6A 7D 0082 769 MOVQ (R10),R1 ;RETRIEVE TABLE NAME DESCRIPTOR
 51 51 3C 0085 770 MOVZWL R1,R1 ;ZERO HIGH ORDER WORD OF LENGTH
 94 13 0088 771 BEQL 940\$;ERROR IF ZERO LENGTH TABLE NAME
 1F 51 B1 008A 772 CMPW R1,#LNMSC_TABNAMLEN ;CHECK SIZE OF TABLE NAME STRING?
 8F 1A 008D 773 BGTRU 940\$;ERROR IF SPECIFIED SIZE EXCEEDS MAXIMUM
 7E 51 7D 0095 774 IFNORD R1,(R2),44\$;CHECK ACCESS TO TABLE NAME BUFFER
 5A 5E D0 0098 775 MOVQ R1,-(SP)
 0098 776 MOVL SP,R10 ;SAVE TABLE NAME DESCRIPTOR
 0098 777 ;ADDRESS OF TABLE NAME DESCRIPTOR

58 1C AC D0 0098 778 40\$: MOVL CTPARTAB(AP),R8 ;ADDRESS OF PARENT TABLE DESCRIPTOR
 29 13 009F 780 BEQL 42\$;ERROR IF NOT PRESENT
 00A1 781 IFNORD #8,(R8),44\$;CHECK ACCESS TO DESCRIPTOR
 50 68 7E 00A7 782 MOVAQ (R8),R0 ;ADDRESS OF DESCRIPTOR
 FF53 30 00AA 783 BSBW LNMS\$PROBER ;PROBE PARENT TABLE NAME STRING
 1D 50 E9 00AD 784 BLBC R0,43\$;BRANCH IF CAN'T READ OR NOT PRESENT
 7E 51 7D 00B0 785 MOVQ R1,-(SP)
 58 5E D0 00B3 786 MOVL SP,R8 ;SAVE PARENT TABLE NAME DESCRIPTOR
 00B6 787 ;ADDRESS OF PARENT TABLE NAME DESCRIPTOR

50 20 AC D0 00B6 788 MOVL CTACMODE(AP),R0 ;GET SPECIFIED ACCESS MODE
 1D 13 00BA 789 BEQL 50\$;BRANCH IF NOT PRESENT
 00BC 790 IFNORD #1,(R0),44\$;CHECK READ ACCESS TO BYTE

50 60 9A 00C2 791 MOVZBL (R0) R0 ;GET VALUE
 50 03 D1 00C5 792 CMPL #PSL\$C_USER,R0 ;CHECK FOR VALID ACCESS MODE
 09 1E 00C8 793 BGEQU 45\$;OKAY
 FF47 31 00CA 794 42\$: BRW 910S ;INVALID ACCESS MODE
 FF4D 31 00CD 795 43\$: BRW 930S ;RETURN
 FF3D 31 00D0 796 44\$: BRW 900S ;ACCESS VIOLATION
 00D3 797
 00D3 798 :
 00D3 799 : IF THE ACCESS MODE OF THE NEW LOGICAL NAME TABLE WAS EXPLICETLY SPECIFIED
 00D3 800 : AND THE CALLER HAS THE SYSNAM PRIVILEGE, THEN THE NEW LOGICAL NAME TABLE IS
 00D3 801 : CREATED WITH THE SPECIFIED ACCESS MODE. OTHERWISE, THE ACCESS MODE OF THE
 00D3 802 : CALLER IS MAXIMIZED WITH ANY EXPLICETLY SPECIFIED ACCESS MODE AND USED TO
 00D3 803 : CREATE THE NEW LOGICAL NAME TABLE.
 00D3 804 :
 00D3 805 :
 00000000'GF 16 00D3 806 45\$: IFPRIV SYSNAM,60S ;SYSNAM REQUIRED TO SPECIFY INNER MODE
 55 50 DD 00D9 807 50\$: JSB G^EXESMAXACMODE ;MAXIMIZE SPECIFIED MODE WITH CALLER'S
 00DF 808 60\$: MOVL R0,R5 ;MODE TO DETERMINE MODE OF NEW TABLE
 00E2 809
 00E2 810 :
 00E2 811 : RAISE IPL TO AST DELIVERY LEVEL SO THAT THERE ARE NO INTERRUPTIONS WHILE
 00E2 812 : THE LOGICAL NAME TABLE ENTRY IS BEING CREATED.
 00E2 813 :
 00E2 814 :
 54 DD 00E2 815 SETIPL S^#IPL\$_ASTDEL ;RAISE TO AST DELIVERY LEVEL
 00E5 816 PUSHL R4 ;SAVE THE PCB ADDRESS
 00E7 817
 00E7 818 :
 00E7 819 : ALLOCATE A LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME TABLE ENTRY. ASSUME
 00E7 820 : THAT THE NEW LOGICAL NAME TABLE WILL BE SHAREABLE, SO ALLOCATE THE SPACE
 00E7 821 : FROM SYSTEM PAGED POOL. UNTIL THE PARENT LOGICAL TABLE BLOCK IS LOCATED,
 00E7 822 : WHETHER THE NEW LOGICAL NAME TABLE WILL BE IN SYSTEM POOL AND SHAREABLE
 00E7 823 : OR IN P1 SPACE AND NOT SHAREABLE IS NOT KNOWN. HOWEVER, THE LOGICAL NAME
 00E7 824 : MUTEX MUST BE LOCKED BEFORE A SEARCH CAN BE MADE FOR THE PARENT LOGICAL NAME
 00E7 825 : TABLE, AND ONCE THE MUTEX IS LOCKED, SYSTEM POOL CAN NOT ALTHOUGH P1 POOL CAN
 00E7 826 : BE ALLOCATED. THEREFORE, THE NEW LOGICAL NAME TABLE ENTRY IS ASSUMED TO BE
 00E7 827 : SHAREABLE, AND IF THE PARENT TABLE IS FOUND TO RESIDE IN SYSTEM POOL, THEN THE
 00E7 828 : SYSTEM POOL ALLOCATED FOR THE NEW ENTRY IS DEALLOCATED, AND PROCESS-PRIVATE
 00E7 829 : POOL IS ALLOCATED IN ITS PLACE.
 00E7 830 :
 00E7 831 :
 5A D5 00E7 832 TSTL R10 ;LOGICAL NAME EXPLICETLY SPECIFIED?
 05 13 00E9 833 BEQL 70S ;NO - GO GET FIXED SIZE TO USE
 51 6A DD 00EB 834 MOVL (R10),R1 ;YES - LOGICAL NAME SIZE IS USED AS
 03 11 00EE 835 BRB 75\$;SIZE OF LNMB NAME
 51 14 DD 00F0 836 70\$: MOVL #LNMB_HEX_TAB_LEN,R1 ;FIXED SIZE QUANTITY
 CO 00F3 837
 CO 00F3 838 75\$: ADDL2 #<- ;LENGTH OF ENTRY =
 00F4 839 LNMB\$T_NAME+1+ - ;SIZE OF LNMB + NAME SIZE COUNT FIELD
 00F4 840 LNMX\$T_XLATION+1+ - ;SIZE OF LNMX + HEADER SIZE COUNT FIELD
 00F4 841 LNMTH\$R_LENGTH+ - ;SIZE OF LNMTH
 00F4 842 ORBSC_LENGTH+ - ;ORB SIZE
 00F4 843 1+ - ;TRAILER BYTE
 00F4 844 ^X07 - ;(ROUND LNMB TO NEXT QUADWORD BOUNDARY)
 51 0000009C 8F 00F4 845 >R1 ;SIZE OF LNMB NAME
 51 07 CA 00FA 846 BICL2 #^X07,R1 ;TRUNCATE LNMB SIZE TO QUADWORD MULTIPLE
 00000000'GF 16 00FD 847 JSB G^EXESALOPAGED ;ALLOCATE THE LOGICAL NAME TABLE ENTRY

03 50 E8 0103 848 BLBS R0,77\$;BRANCH IF SUFFICIENT MEMORY
 08 A2 51 B0 0109 849 77\$: BRW 920\$;OTHERWISE RETURN ERROR
 010D 850 :MOVW R1,LNMB\$W_SIZE(R2) ;SAVE NEW BLOCK SIZE WITHIN THE BLOCK

010D 851 :
 010D 852 :
 010D 853 : WRITE LOCK THE LOGICAL NAME MUTEX AND THEN POSITION TO THE PARENT LOGICAL
 010D 854 : NAME TABLE. DEALLOCATE THE NEW LNMB IF UNABLE TO POSITION TO THE PARENT
 010D 855 : TABLE ENTRY.
 010D 856 :
 010D 857 :
 52 DD 010D 858 PUSHL R2 ;SAVE LOGICAL NAME BLOCK ADDRESS
 51 FEEF 30 010F 859 BSBW LNMSLOCKW ;LOCK TABLES FOR WRITING
 52 55 D0 0112 860 MOVL R5,R1 ;ACCESS MODE
 52 68 7D 0115 861 MOVQ (R8),R2 ;TABLE NAME DESCRIPTOR
 52 FEE5 30 0118 862 BSBW LNMSFIRSTTAB ;SEARCH FOR SPECIFIED PARENT TABLE
 03 50 E8 011B 863 POPL R4 ;RESTORE LOGICAL NAME BLOCK ADDRESS
 01AC 31 0121 864 BLBS R0,80\$;BRANCH ON FAILURE
 0124 865 BRW 160\$
 0124 866 :
 0124 867 : IF THE PARENT LOGICAL NAME TABLE IS SHAREABLE THEN:
 0124 868 :
 0124 869 : 1. MARK THE NEW TABLE AS SHAREABLE.
 0124 870 : 2. MAKE SURE THE NEW TABLE IS NOT MARKED AS CONFINED.
 0124 871 : 3. MAKE SURE THE CALLER HAS QUOTA ACCESS TO THE PARENT LOGICAL NAME TABLE.
 0124 872 : 4. MAKE SURE THAT THE CALLER HAS WRITE ACCESS TO THE SYSTEM DIRECTORY TABLE
 0124 873 : PROVIDED THE USER IS SPECIFYING THE NAME OF THE SHAREABLE TABLE AND NOT
 0124 874 : LETTING VMS DEFAULT NAME IT.
 0124 875 :
 0124 876 :
 0124 877 :
 3B 00 E1 0124 878 80\$: BBC #LNMT\$V_SHAREABLE - ;BRANCH IF THE PARENT TABLE ENTRY IS IN
 00 57 61 0126 879 LNMTH\$B_FLAGS(R1),90\$;PROCESS-PRIVATE SPACE
 00 57 02 CA 0128 880 BICL2 #LNMSM_CONFINE,R7 ;SHAREABLE TABLES CAN'T BE CONFINED
 00 57 10 E3 012B 881 BBCS #LNMSV_SHAREABLE,R7,83\$;MARK NEW TABLE AS SHAREABLE
 012F 882 :
 52 04 9A 012F 883 83\$: MOVZBL #QUOTA_ACCESS,R2 ;CODE FOR ACCESS CHECK
 54 54 DD 0132 884 PUSHL R4 ;SAVE LNMB PTR
 54 04 AE D0 0134 885 MOVL 4(SP),R4 ;RESTORE PCB
 54 FEC5 30 0138 886 BSBW LNMSCHECK_PROT ;CHECK THE PROTECTION
 54 8E D0 013B 887 MOVL (SP)+,R4 ;RESTORE LNMB
 03 50 E8 013E 888 BLBS R0,85\$;EXIT ON FAILURE
 018C 31 0141 889 BRW 160\$
 0144 890 :
 5A D5 0144 891 85\$: TSTL R10 ;WAS A NAME SPECIFIED FOR THE NEW TABLE
 59 13 0146 892 BEQL 100\$;BRANCH IF ONE WASN'T
 0148 893 :
 52 02 9A 0148 894 MOVZBL #WRITE_ACCESS,R2 ;CODE FOR ACCESS CHECK
 51 12 BB 0148 895 PUSHR #^M<R1,R4> ;SAVE LNMB AND PARENT TABLE HEADER ADDR
 51 09 A1 D0 014D 896 MOVL LNMTH\$L_NAME(R1),R1 ;RETRIEVE ADDRESS OF PARENT LNMB
 51 OC A1 D0 0151 897 MOVL LNMB\$L_TABLE(R1),R1 ;RETRIEVE SYSTEM DIRECTORY TABLE HEADER
 54 08 AE D0 0155 898 MOVL 8(SP),R4 ;RESTORE PCB
 54 FEA4 30 0159 899 BSBW LNMSCHECK_PROT ;CHECK THE PROTECTION
 12 BA 015C 900 POPR #^M<R1,R4> ;RESTORE LNMB AND PARENT TABLE HEADER
 70 50 E9 015E 901 BLBC R0,105\$;EXIT ON FAILURE
 3E 11 0161 902 BRB 100\$
 0163 903 :
 0163 904 :

0163 905 : IF THE PARENT TABLE IS DISCOVERED TO EXIST IN PROCESS-PRIVATE SPACE, THEN
 0163 906 : THIS IS ALSO WHERE THE NEW LOGICAL NAME TABLE ENTRY SHOULD GO. DEALLOCATE
 0163 907 : THE SYSTEM SPACE LNMB, AND ALLOCATE A SUFFICIENTLY SIZED BLOCK FROM P1
 0163 908 : SPACE.
 0163 909 :
 0163 910 :

6E 7E 08 51 DD 0163 911 90\$: PUSHL R1
 00000058 A4 3C 0165 912 MOVZWL LNMB\$W_SIZE(R4),-(SP)
 50 54 DO 0169 913 SUBL2 #ORBSK_LENGTH,(SP)
 FE8A' 30 0170 914 MOVL R4,RO
 0173 915 BSBW LNMSDELBLK
 0176 916
 00000000' GF 51 8ED0 0176 917 POPL R1
 08 50 E8 0179 918 JSB G^EXESALOP1PROC
 0124 8F 3C 0182 919 BLBS R0,95\$
 0186 31 0187 920 MOVZWL #SSS_INSFMEM,(SP)
 018A 921 BRW 190\$
 08 A4 54 52 DO 018A 923 95\$: MOVL R2,R4
 51 8ED0 018D 924 MOVW R1,LNMB\$W_SIZE(R4)
 52 09 A1 0191 925 POPL R1
 01 E1 0194 926 MOVL LNMTHSL_NAME(R1),R2
 04 10 A2 0198 927 BBC #LNMB\$V_CONFINE -
 00 57 01 E3 019D 928 LNMBSB_FLAGS(R2),100\$
 01A1 929 BBCS #LNMSV_CONFINE,R7,100\$
 01A1 930
 01A1 931
 01A1 932 : PERFORM QUOTA CHECKS. TWO SUCH QUOTA CHECKS ARE MADE:
 01A1 933
 01A1 934 : 1. THE CONTAINING TABLE (SYSTEM OR PROCESS DIRECTORY TABLE) IS CHECKED TO
 01A1 935 : MAKE SURE IT HAS SUFFICIENT QUOTA TO CONTAIN THE NEW LOGICAL NAME TABLE
 01A1 936 : ENTRY. NOTE THAT AN ASSUMPTION IS MADE THAT THE DIRECTORY TABLES ARE
 01A1 937 : THEIR OWN QUOTA HOLDERS.
 01A1 938
 01A1 939 : 2. THE PARENT LOGICAL NAME TABLE'S QUOTA HOLDER IS CHECKED TO MAKE SURE IT HAS
 01A1 940 : SUFFICIENT QUOTA FOR THE QUOTA WHICH WILL BE DEDUCTED FROM IT AND ALLOCATED
 01A1 941 : SPECIFICALLY TO THE NEW TABLE.
 01A1 942 :
 01A1 943
 52 09 A1 DO 01A1 944 100\$: MOVL LNMTHSL_NAME(R1),R2
 52 0C A2 DO 01A5 945 MOVL LNMB\$L_TABLE(R2),R2
 50 08 A4 3C 01A9 946 MOVZWL LNMB\$W_SIZE(R4),R0
 21 A2 50 D1 01AD 947 CMPL R0,LNMTHSL_BYTES(R2)
 19 1A 01B1 948 BGTRU 103\$
 58 10 AC DO 01B3 949
 20 13 01B7 950 MOVL CTQUOTA(AP),R8
 58 68 DO 01B9 951 BEQL 110\$
 58 68 DO 01BF 952 IFNORD #4,(R8),107\$
 01C2 953 MOVL (R8),R8
 52 19 A1 DO 01C2 954
 21 A2 58 D1 01C6 955 101\$: MOVL LNMTHSL_QTABLE(R1),R2
 0D 1B 01CA 956 CMPL R8,LNMTRSL_BYTES(R2)
 50 224C 8F 3C 01CC 957 BLEQU 110\$
 00FC 31 01D1 958 103\$: MOVZWL #SSS_EXLNMQOUTA,RO
 01D4 959 105\$: BRW 160\$
 50 0C 3C 01D4 960
 01D4 961 107\$: MOVZWL #SSS_ACCVIO,RO
 :NO ACCESS

F8 11 01D7 962 BRB 105\$;EXIT

01D9 963
01D9 964
01D9 965 : FILL IN THE LNMB PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME
01D9 966 TABLE ENTRY.
01D9 967
01D9 968 : REGISTER USAGE IS AS FOLLOWS:
01D9 969
01D9 970 R1 = ADDRESS OF TABLE HEADER FOR PARENT LOGICAL NAME TABLE.
01D9 971 R3 = SIZE OF NEW LOGICAL NAME TABLE ENTRY'S NAME.
01D9 972 R4 = ADDRESS OF LOGICAL NAME BLOCK FOR NEW LOGICAL NAME TABLE.
01D9 973 R5 = ACCESS MODE.
01D9 974 R6 = ADDRESS OF PROTECTION MASK.
01D9 975 R7 = ATTRIBUTE BITS.
01D9 976 R8 = QUOTA.
01D9 977 R9 = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
01D9 978 R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
01D9 979 R11 = ADDRESS OF WORD TO RECEIVE TABLE NAME LENGTH.
01D9 980
01D9 981

50 0A A4 9E 01D9 982 110\$: MOVAB LNMB\$B_TYPE(R4),R0 ;POSITION TO BLOCK TYPE
80 40 8F 90 01DD 983 MOVB #DYNSC_LNM,(R0)+ ;SET DATA STRUCTURE TYPE
80 55 90 01E1 984 MOVB R5,(R0)+ ;SET OWNER ACCESS MODE
52 09 A1 D0 01E4 985 MOVL LNMT\$L_NAME(R1),R2 ;ADDRESS OF PARENT'S LOGICAL NAME BLOCK
80 0C A2 D0 01E8 986 MOVL LNMB\$L_TABLE(R2),(R0)+ ;STORE DIRECTORY TABLE HEADER ADDRESS
80 08 57 89 01EC 987 BISB3 R7,#LNMB\$M_TABLE,(R0)+ ;STORE NAME ATTRIBUTES
01F0 990
01F0 991 :
01F0 992 : IF THE CALLER HAS SPECIFIED A LOGICAL NAME STRING THEN THAT STRING BECOMES
01F0 993 : THE NAME OF THE LNMB FOR THE NEW LOGICAL NAME TABLE ENTRY BY MOVING IT INTO
01F0 994 : THE LOGICAL NAME BLOCK. OTHERWISE, THE UNIQUE NAME OF THE LNMB IS A
01F0 995 : CONSTRUCTED QUANTITY OF FIXED (12 BYTES) SIZE.
01F0 996 :
01F0 997 :
5A D5 01F0 998 TSTL R10 ;LOGICAL NAME STRING SPECIFIED?
2A 13 01F2 999 BEQL 120\$;NO - BRANCH TO CONSTRUCT UNIQUE NAME
80 6A 90 01F4 1000 MOVBL (R10), (R0)+ ;STORE LENGTH OF NAME
7E 54 7D 01F7 1001 MOVQ R4,-(SP) ;SAVE NEEDED REGISTERS OVER MOVTUC
51 DD 01FA 1002 PUSHL R1
00 04 BA 6A 2F 01FC 1003 MOVTUC (R10),@4(R10),#0,- ;CHECK FOR PROPER TABLE NAME SYNTAX AND
00000000'EF 0201 1004 EXESLNM_SYNTAX_DAT,- ;AT THE SAME TIME MOVE THE TABLE NAME
60 6A 0206 1005 (R10),(R0) ;INTO THE LOGICAL NAME TABLE ENTRY
0B 1D 0208 1006 BVS 115\$;BRANCH IF TABLE NAME HAS INVALID CHARS
50 55 020A 1007 MOVL R5,R0 ;POSITION PAST LNMB NAME STRING
51 8ED0 020D 1008 POPL R1 ;RESTORE SAVED REGISTERS
54 8E 7D 0210 1009 MOVQ (SP)+,R4
46 11 0213 1010 BRB 130\$;BRANCH AROUND UNIQUE NAME CONSTRUCTION
50 015C 32 BA 0215 1012 115\$: POPR #^M<R1,R4,R5> ;RESTORE REGISTERS
8F 3C 0217 1013 MOVZWL #SSS_IVLOGTAB,R0 ;INVALID TABLE NAME STRING SPECIFIED
B3 11 021C 1014 BRB 105\$;SO GO RETURN THE APPROPRIATE ERROR
80 244D4E4C 80 14 90 021E 1016 120\$: MOVB #LNM_HEX_TAB_LEN,(R0)+ ;STORE LENGTH OF NAME
8F DO 0221 1017 MOVL #^A/[NMS7,(R0)+ ;STORE BEGINNING OF NAME
53 1C DO 0228 1018 MOVL #<8-1>*4,R3 ;STARTING NIBBLE POSITION

52 54 04 53 EF 022B 1019 125\$: EXTZV R3,#4,R4,R2 :
 80 FDCB CF42 90 0230 1020 MOVB HEXDIGITS[R2],(R0)+ ;
 53 04 C2 0236 1021 SUBL #4,R3 ;
 53 04 F0 18 0239 1022 BGEQ 125\$;
 54 DD 023B 1023 PUSHL R4 ;
 54 04 AE DO 023D 1024 MOVL 4(SP),R4 ;
 54 64 A4 DO 0241 1025 MOVL PCB\$L EPID(R4),R4 ;
 52 54 53 1C DO 0245 1026 MOVL #C8-15*4,R3 ;
 80 FDAE CF42 90 024D 1028 EXTZV R3,#4,R4,R2 ;
 53 04 C2 0253 1029 MOVB HEXDIGITS[R2],(R0)+ ;
 53 04 F0 18 0256 1030 SUBL #4,R3 ;
 54 8E DO 0258 1031 BGEQ 126\$;
 54 8E DO 025B 1032 MOVL (SP)+,R4 ;
 025B 1033 : FILL IN THE LNMX PORTION OF THE LOGICAL NAME
 025B 1034 : TABLE ENTRY. FOR LOGICAL NAME TABLES THE SOLE TRANSLATION CONSISTS OF THE
 025B 1035 : TABLE HEADER.
 025B 1036 :
 025B 1037 :
 80 80 02 90 025B 1038 130\$: MOVBL #LNMX\$M_TERMINAL,(R0)+ ;
 80 82 8F 90 025E 1039 MOVBL #LNMX\$C_TABLE,(R0)+ ;
 80 80 B4 0262 1040 CLRW (R0)+ ;
 80 25 90 0264 1041 MOVBL #LNMTHSK_LENGTH,(R0)+ ;
 0267 1042 :
 0267 1043 : FILL IN THE LNMTH PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME
 0267 1044 : TABLE ENTRY. THE TABLE HEADER OF THE NEW ENTRY CONSTITUTES THE SOLE
 0267 1045 : TRANSLATION OF THE NEW LOGICAL NAME TABLE ENTRY.
 0267 1046 :
 0267 1047 :
 0267 1048 :
 53 52 50 D0 0267 1049 MOVL R0,R2 ;
 53 57 10 9C 026A 1050 ROTL #16,R7,R3 ;
 80 53 90 026E 1051 MOVB R3,(R0)+ ;
 0271 1052 :
 80 01 A1 DO 0271 1053 MOVL LNMTHS\$L_HASH(R1),(R0)+ ;
 0275 1054 :
 80 80 D4 0275 1055 CLRL (R0)+ ;
 80 64 9E 0277 1056 MOVAB (R4),(R0)+ ;
 80 51 D0 027A 1057 MOVL R1,(R0)+ ;
 80 7C 027D 1058 CLRQ (R0)+ ;
 027F 1059 :
 027F 1060 :
 80 19 A1 DO 027F 1061 MOVL LNMTHS\$L_QTABLE(R1),(R0)+ ;
 60 58 DO 0283 1062 MOVL R8,(R0) ;
 04 13 0286 1063 BEQL 135\$;
 19 A2 52 DO 0288 1064 MOVL R2,LNMTHS\$L_QTABLE(R2) ;
 80 80 DO 028C 1065 135\$: MOVL (R0)+,(R0)+ ;
 028F 1066 :
 028F 1067 :
 028F 1068 : FILL IN THE NEXT LNMX PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL
 028F 1069 : NAME TABLE ENTRY. THE LAST TRANSLATION BLOCK CONSISTS SOLELY OF A FLAGS
 028F 1070 : FIELD, AND IS MARKED WITHIN THIS FLAGS FIELD AS THE LAST LNMX.
 028F 1071 :
 80 04 90 028F 1072 MOVB #LNMX\$M_XEND,(R0)+ ;
 0292 1073 :
 0292 1074 :
 0292 1075 :

0292 1076 : INITIALIZE THE OBJECT RIGHTS BLOCK WHICH EXISTS FOR ALL SHAREABLE LOGICAL NAME
 0292 1077 : TABLES. PROCESS-PRIVATE LOGICAL NAME TABLES DO NOT REQUIRE SUCH A ORB BECAUSE
 0292 1078 : BECAUSE PROTECTION CHECKING IS NOT REQUIRED, AND IS NEVER PERFORMED ON THEM.
 0292 1079 :
 0292 1080 :
 03 00 E1 0292 1081 BBC #LNMB\$V_SHAREABLE_-
 FD67' 30 0294 1082 LNMTHSB_FLAGS(R1),137\$;IS PARENT LOGICAL NAME TABLE SHAREABLE?
 56 52 D0 0296 1083 BSBW LNMTHSL_NAME(R1),R2 ;IF NOT, SKIP INITIALIZATION OF ORB
 0299 1084 137\$: MOVL R2,R6 ;INITIALIZE TABLE'S OBJECT RIGHTS BLOCK
 029C 1085 ;SAVE TABLE HEADER ADDRESS
 029C 1086 :
 029C 1087 : SET THE TEMPORARY BIT LNMB\$V_NODELETE IN THE FLAGS BYTE OF THE LNMB BLOCK OF
 029C 1088 : THE NEW LOGICAL NAME TABLE ENTRY'S PARENT, GRANDPARENT ETC... THIS WILL
 029C 1089 : PREVENT THE NEW ENTRY FROM BEING INSERTED IF ITS INSERTION WOULD RESULT IN
 029C 1090 : THE DELETION OF ANY LOGICAL NAME TABLE ENTRY IN ITS DIRECT LINE OF DESCENT;
 029C 1091 : AND THUS, THE DELETION OF ITS PARENT TABLE.
 029C 1092 :
 029C 1093 :
 52 01 E0 029C 1094 140\$: BBS #LNMB\$V_DIRECTORY_-
 0E 61 029E 1095 LNMTHSB_FLAGS(R1),150\$;SYSTEM/PROCESS DIRECTORY TABLE?
 09 A1 D0 02A0 1096 MOVL LNMTHSL_NAME(R1),R2 ;YES - GO INSERT NEW TABLE ENTRY
 10 88 02A4 1097 BISB2 #LNMB\$M_NODELETE,- ;ADDRESS OF PARENT'S LNMB BLOCK
 10 A2 02A6 1098 MOVL LNMB\$B_FLAGS(R2) ;SET THE TEMPORARY BIT BLOCKING DELETION
 51 0D A1 D0 02A8 1099 LNMTHSC_PARENT(R1),R1 ;OF THIS LOGICAL NAME TABLE ENTRY
 EE 11 02AC 1100 BRB 140\$;NO - RETRIEVE PARENT'S TABLE HEADER
 02AE 1101 ; ADDRESS AND CONTINUE
 02AE 1102 :
 02AE 1103 : AT THIS POINT ALL CHECKS HAVE BEEN MADE, AND THE TABLE STRUCTURES MAY NOW
 02AE 1104 : BE MODIFIED FOR THE FIRST TIME. INSERT THE NEW LOGICAL NAME TABLE ENTRY AND
 02AE 1105 : THEN CLEAR THE TEMPORARY DO NOT DELETE BIT, LNMB\$V_NODELETE REGARDLESS OF THE
 02AE 1106 : OUTCOME OF THE INSERTION.
 02AE 1107 :
 02AE 1108 : THIS INSERTION MAY TAKE ONE OF THREE FORMS:
 02AE 1109 :
 02AE 1110 : 1. THE NEW LOGICAL NAME TABLE ENTRY MAYBE INSERTED AS A NEW LOGICAL NAME
 02AE 1111 : TABLE. FOR THIS CASE TO OCCUR THE FOLLOWING CONDITIONS MUST HOLD TRUE.
 02AE 1112 :
 02AE 1113 : A. THERE IS NO EXISTING LOGICAL NAME TABLE ENTRY IN THE SAME DIRECTORY
 02AE 1114 : (PROCESS OR SYSTEM) WITH THE SAME NAME AND ACCESS MODE.
 02AE 1115 :
 02AE 1116 : B. THERE IS NO EXISTING LOGICAL NAME TABLE ENTRY IN THE SAME DIRECTORY
 02AE 1117 : (PROCESS OR SYSTEM) WITH THE SAME NAME AND AN INNER ACCESS MODE THAT
 02AE 1118 : DOES NOT ALLOW ALIASES.
 02AE 1119 :
 02AE 1120 : 2. THE LOGICAL NAME TABLE ENTRY MAYBE "MAPPED" INTO AN EXISTING LOGICAL NAME
 02AE 1121 : TABLE ENTRY. FOR THIS CASE TO OCCUR THE FOLLOWING CONDITIONS MUST BE MET.
 02AE 1122 :
 02AE 1123 : A. THE LNMSV_CREATE_IF BIT MUST HAVE BEEN SET IN THE ATTRIBUTES FIELD.
 02AE 1124 :
 02AE 1125 : B. THERE MUST EXIST A LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AND
 02AE 1126 : ACCESS MODE WITHIN THE SAME DIRECTORY (PROCESS OR SYSTEM).
 02AE 1127 :
 02AE 1128 : 3. THE LOGICAL NAME TABLE ENTRY MAYBE INSERTED SUPERSEDING AN EXISTING ENTRY
 02AE 1129 : IN THE SAME DIRECTORY (PROCESS OR SYSTEM) WITH THE SAME NAME AND ACCESS
 02AE 1130 : MODE. THE OLD LOGICAL NAME TABLE ENTRY TOGETHER WITH ITS ENTIRE HIERARCHY
 02AE 1131 : OF LOGICAL NAMES AND LOGICAL NAME TABLE ENTRIES IS DELETED, PROVIDED OF
 02AE 1132 : COURSE, THAT THE OLD ENTRY WAS FOR A LOGICAL NAME TABLE. THE CONDITIONS

02AE 1133 : WHICH MUST BE MET FOR THIS CASE TO OCCUR ARE:
 02AE 1134
 02AE 1135 A. THE LNMSV_CREATE_IF BIT MUST NOT HAVE BEEN SET IN THE ATTRIBUTES FIELD.
 02AE 1136
 02AE 1137 B. THERE MUST EXIST A LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AND
 02AE 1138 ACCESS MODE WITHIN THE SAME DIRECTORY (PROCESS OR SYSTEM).
 02AE 1139
 02AE 1140 IF A CASE 1 OR 3 INSERTION IS PERFORMED, AND THE NEW LOGICAL NAME TABLE ENTRY
 02AE 1141 DOES NOT ALLOW ALIASES, THEN ANY LOGICAL NAME TABLE ENTRIES IN THE SAME
 02AE 1142 DIRECTORY (PROCESS OR SYSTEM) WITH THE SAME NAME BUT AT AN OUTER ACCESS MODES
 02AE 1143 ARE DELETED ALONG WITH THEIR HIERARCHY OF LOGICAL NAMES AND LOGICAL NAME
 02AE 1144 TABLE ENTRIES, PROVIDED THEY TOO ARE LOGICAL NAME TABLES.
 02AE 1145
 02AE 1146

51 54 DO	02AE 1147	150\$: MOVL R4,R1	:ADDRESS OF NEW LOGICAL NAME TABLE ENTRY
52 57 DO	02B1 1148	MOVL R7,R2	:ATTRIBUTES
FD49' 30	02B4 1149	BSBW LNMSINSLOGTAB	:INSERT NEW LOGICAL NAME TABLE ENTRY
53 51 DO	02B7 1150	MOVL R1,R3	:SAVE LNMB ADDRESS OF MAPPED-TO ENTRY
	02BA 1151		:IF SUCH A MAPPING HAS TAKEN PLACE
	02BA 1152		
51 0D A6 DO	02BA 1153	MOVL LNMTH\$L_PARENT(R6),R1	:RETRIEVE PARENT'S TABLE HEADER
01 E0	02BE 1154	BBS #LNMTTH\$V DIRECTORY,-	:SYSTEM/PROCESS DIRECTORY TABLE?
0E 61	02C0 1155	LNMTH\$B_FLAGS(R1),160\$:YES - GO CHECK INSERTION
52 09 A1 DO	02C2 1156	MOVL LNMTH\$L_NAME(R1),R2	:ADDRESS OF PARENT'S LNMB BLOCK
10 8A	02C6 1157	BICB2 #LNMB\$M_NODELETE,-	:CLEAR THE TEMPORARY BIT BLOCKING
10 A2	02C8 1158	LNMBS\$B_FLAGS(R2)	:DELETION OF THIS LNMB
51 0D A1 DO	02CA 1159	MOVL LNMTH\$C_PARENT(R1),R1	:NO - RETRIEVE PARENT'S TABLE HEADER
EE 11	02CE 1160	BRB 155\$: ADDRESS AND CONTINUE
	02DD 1161		
08 50 DD	02DD 1162	160\$: PUSHL RO	:SAVE STATUS
08 50 E8	02D2 1163	BLBS RO,170\$:BRANCH ON SUCCESS
50 54 DO	02D5 1164	MOVL R4,RO	:LOGICAL NAME BLOCK FOR NEW TABLE ENTRY
FD25' 30	02D8 1165	BSBW LNMSDELBLK	:DELETE BLOCK CONTAINING NEW TABLE ENTRY
33 11	02DB 1166	BRB 190\$:JOIN MAIN EXIT - STATUS ON STACK
	02DD 1167		
	02DD 1168		
	02DD 1169	: AT THIS POINT THE NEW LOGICAL NAME TABLE HAS SUCCESSFULLY BEEN INSERTED.	
	02DD 1170	: FILL IN ANY SPECIFIED OUTPUT PARAMETERS (TABLEN AND/OR TABNAM) BEFORE	
	02DD 1171	: RETURNING THE STATUS TO THE CALLER OF THE SYSTEM SERVICE. IF THE NEW TABLE	
	02DD 1172	: ENTRY WAS JUST MAPPED TO AN EXISTING TABLE ENTRY, THEN THE LOGICAL NAME BLOCK	
	02DD 1173	: CONTAINING THE NEW TABLE ENTRY IS DELETED BEFORE RETURNING.	
	02DD 1174		
	02DD 1175		
50 01 D1	02DD 1176	170\$: CMPL #SSS_NORMAL,RO	: WAS NEW ENTRY MAPPED TO EXISTING ONE?
0B 12	02E0 1177	BNEQ 175\$: BRANCH IF NOT
50 54 DO	02E2 1178	MOVL R4,RO	: ADDRESS OF NEW ENTRY
53 DD	02E5 1179	PUSHL R3	: SAVE ADDRESS OF ENTRY MAPPED TO
FD16' 30	02E7 1180	BSBW LNMSDELBLK	: DELETE NEW ENTRY
53 8EDO	02EA 1181	POPL R3	: ADDRESS OF ENTRY MAPPED TO
	02ED 1182		
51 11 A4 9E	02ED 1183	175\$: MOVAB LNMB\$T_NAME(R4),R1	: ADDRESS OF TABLE NAME COUNTED STRING
50 81 9A	02F1 1184	MOVZBL (R1)+,RO	: LENGTH OF NAME STRING
	02F4 1185		
	02F4 1186	TSTL R11	: RETURN LENGTH?
5B D5	02F6 1187	BEQL 180\$: BRANCH IF NO
03 13	02F8 1188	MOVW RO,(R11)	: RETURN LENGTH
6B 50 B0	02FB 1189		

	59	D5	02FB	1190	180\$:	TSTL	R9	:WANT LOGICAL NAME TABLE NAME RETURNED?
	11	13	02FD	1191		BEQL	190\$:BRANCH IF NO
	69	50	B1	02FF	1192	CMPW	R0 (R9)	:ENOUGH ROOM?
	07	14	0302	1193		BGTR	185\$:BRANCH IF NOT ENOUGH ROOM
04 B9	61	50	28	0304	1194	MOV C3	R0 (R1),@4(R9)	:RETURN NAME STRING
	05	11	0309	1195		BRB	190\$:NOT ENOUGH ROOM FOR NAME
6E	0214	8F	3C	030B	1196	185\$:	MOVZWL	#SSS_RESULTOVF,(SP)
				0310	1197			
54	04 AE,	D0	0310	1198	190\$:	MOVL	4(SP),R4	:RETRIEVE PCB ADDRESS
	FCE9,	30	0314	1199		BSBW	LNM\$UNLOCK	:UNLOCK TABLES
	50 8ED0		0317	1200		POPL	R0	:RESTORE STATUS
	04	031A	1201			RET		:RETURN TO CALLER
			031B	1202				
			031B	1203		:DISABLE	LSB	
			031B	1204	:	:PAGE		

031B 1206 .SBTTL EXE\$CRELNM - CREATE LOGICAL NAME
031B 1207 +
031B 1208 EXE\$CRELNM - CREATE LOGICAL NAME
031B 1209
031B 1210 THIS SERVICE PROVIDES THE CAPABILITY TO CREATE A LOGICAL NAME ENTRY WITH
031B 1211 SOME NUMBER OF EQUIVALENCE STRINGS.
031B 1212
031B 1213 INPUTS:
031B 1214
031B 1215 CNATTR(AP) = ADDRESS OF LOGICAL NAME ATTRIBUTES.
031B 1216 CNTABNAM(AP) = ADDRESS OF TABLE NAME STRING DESCRIPTOR.
031B 1217 CNLOGNAM(AP) = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR.
031B 1218 CNACMODE(AP) = ADDRESS OF ACCESS MODE OF LOGICAL NAME TO BE CREATED.
031B 1219 CNITMLST(AP) = ADDRESS OF ITEM LIST DEFINING TRANSLATIONS.
031B 1220
031B 1221 R4 = CURRENT PROCESS PCB ADDRESS.
031B 1222
031B 1223 OUTPUTS:
031B 1224
031B 1225 R0 LOW BIT CLEAR INDICATES FAILURE TO CREATE LOGICAL NAME TABLE ENTRY.
031B 1226
031B 1227 R0 = SSS_ACCVIO - LOGICAL NAME DESCRIPTOR, LOGICAL NAME STRING,
031B 1228 TABLE NAME DESCRIPTOR, TABLE NAME STRING, AN ITEM IN THE
031B 1229 ITEM LIST, AN INDEX ITEM BUFFER, AN ATTRIBUTES ITEM
031B 1230 BUFFER, A STRING ITEM BUFFER, CANNOT BE READ BY CALLING
031B 1231 ACCESS MODE. TABLE ITEM BUFFER, TABLE ITEM SIZE BUFFER
031B 1232 CANNOT BE WRITTEN BY CALLING ACCESS MODE.
031B 1233
031B 1234 R0 = SSS_BADPARAM - INVALID ATTRIBUTE, ACCESS MODE, ITEM TYPE,
031B 1235 ITEM LENGTH, ITEM SPECIFIED. LOGICAL NAME DESCRIPTOR,
031B 1236 TABLE NAME DESCRIPTOR NOT SPECIFIED.
031B 1237
031B 1238 R0 = SSS_DUPLNAM - ATTEMPT MADE TO SUPERSEDE NON-ALIASABLE
031B 1239 LOGICAL NAME TABLE ENTRY.
031B 1240
031B 1241 R0 = SSS_EXLNMQUOTA - INSUFFICIENT QUOTA AVAILABLE IN THE QUOTA
031B 1242 TABLE FOR THE CREATION OF THE NEW LOGICAL NAME TABLE
031B 1243 ENTRY.
031B 1244
031B 1245 R0 = SSS_INSFMEM - SUFFICIENT SYSTEM DYNAMIC MEMORY DOES NOT
031B 1246 EXIST TO ALLOCATE LOGICAL NAME TABLE ENTRY AND
031B 1247 IMPLICIT RESOURCE WAIT IS NOT ENABLED.
031B 1248
031B 1249 R0 = SSS_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
031B 1250 LOGICAL NAME STRING, TABLE NAME STRING, OR EQUIVALENCE
031B 1251 STRING SPECIFIED, OR THE LOGICAL NAME IS TO BE CONTAINED
031B 1252 WITHIN A DIRECTORY TABLE AND IS EITHER GREATER THAN 31
031B 1253 CHARACTERS IN SIZE OR HAS INVALID CHARACTERS FOR SUCH
031B 1254 NAMES.
031B 1255
031B 1256 R0 = SSS_IVLOGTAB - INVALID LOGICAL NAME TABLE NAME SPECIFIED.
031B 1257
031B 1258 R0 = SSS_NOLOGTAB - LOGICAL NAME TABLE SPECIFIED DOES NOT EXIST.
031B 1259
031B 1260 R0 = SSS_TOOMANYLNM - TOO MANY LEVELS OF RECURSION IN SEARCH
031B 1261 FOR LOGICAL NAME TABLE.
031B 1262

031B 1263 : RO LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
 031B 1264 :
 031B 1265 : RO = SSS BUFFEROVF - REQUEST SUCCESSFULLY COMPLETED. AN ITEM
 031B 1266 : BUFFER IS NOT LARGE ENOUGH TO HOLD REQUESTED DATA.
 031B 1267 :
 031B 1268 : RO = SSS NORMAL - NORMAL COMPLETION, NEW ENTRY ENTERED IN
 031B 1269 : SPECIFIED LOGICAL NAME TABLE.
 031B 1270 :
 031B 1271 : RO = SSS SUPERSEDE - NORMAL COMPLETION, NEW ENTRY SUPERSEDED
 031B 1272 : PREVIOUS ENTRY IN SPECIFIED LOGICAL NAME TABLE.
 031B 1273 :
 031B 1274 : SIDE EFFECTS:
 031B 1275 :
 031B 1276 : THIS ROUTINE EXITS AT IPL 2, AND MUST CONTINUE TO DO SO, BECAUSE
 031B 1277 : IT IS CALLED AT SYSTEM INITIALIZATION TIME.
 031B 1278 :
 031B 1279 :-
 031B 1280 :
 031F' 00000005 1281 .PSECT Y\$EXEPAGED
 OFFC 0005 1282 .ENTRY EXESCRELMN,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
 31 0007 1283 BRW EXECRELMN
 000A 1284 :
 0000031B 1285 .PSECT YF\$SLNM
 031B 1286 .ENABLE LSB
 031B 1287 :
 031B 1288 :
 031B 1289 : ERROR MESSAGES AND RETURNS
 031B 1290 :
 031B 1291 :
 50 0C 3C 031B 1292 9000\$: MOVZWL #SSS_ACCVIO,RO ;ACCESS VIOLATION
 04 031E 1293 RET
 50 14 3C 031F 1294 9010\$: MOVZWL #SSS_BADPARAM,RO ;BAD SYSTEM SERVICE PARAMETER
 04 0322 1295 RET
 50 0124 8F 3C 0323 1296 9020\$: MOVZWL #SSS_INSFFMEM,RO ;INSUFFICIENT MEMORY
 04 0328 1297 9030\$: RET
 0329 1298 :
 0329 1299 EXECRELMN:
 0329 1300 :
 0329 1301 :
 0329 1302 : VALIDATE AND COPY PARAMETERS AS NECESSARY. REGISTER ASSIGNMENT FOR THE
 0329 1303 : PARAMETERS ARE AS FOLLOWS:
 0329 1304 :
 0329 1305 : R5 = ACCESS MODE
 0329 1306 : R7 = ATTRIBUTE BITS.
 0329 1307 : R9 = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
 0329 1308 : R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
 0329 1309 : R11 = ADDRESS OF START OF ITEM LIST.
 0329 1310 :
 0329 1311 :---
 0329 1312 :
 50 10 AC D0 0329 1313 MOVL CNACMODE(AP),RO :GET SPECIFIED ACCESS MODE
 14 13 0320 1314 BEQL 50\$:BRANCH IF NOT PRESENT
 50 60 9A 032F 1315 IFNORD #1,(R0),9000\$:CHECK READ ACCESS TO BYTE
 50 03 D1 0335 1316 MOVZBL (R0),RO :GET VALUE
 E2 1F 0338 1317 CMPL #PSL\$C_USER,RO :CHECK FOR VALID ACCESS MODE
 033B 1318 BLSSU 9010\$:INVALID ACCESS MODE
 033D 1319 :

033D 1320 :
 033D 1321 : IF THE ACCESS MODE OF THE NEW LOGICAL NAME WAS EXPLICITLY SPECIFIED AND THE
 033D 1322 : CALLER HAS THE SYSNAM PRIVILEGE, THEN THE NEW LOGICAL NAME IS CREATED WITH
 033D 1323 : THE SPECIFIED ACCESS MODE. OTHERWISE, THE ACCESS MODE OF THE CALLER IS
 033D 1324 : MAXIMIZED WITH ANY EXPLICITLY SPECIFIED ACCESS MODE AND USED TO CREATE THE
 033D 1325 : NEW LOGICAL NAME.
 033D 1326 :
 033D 1327 :
 033D 1328 : IFPRIV SYSNAM,60\$
 00000000'GF 16 0343 1329 50\$: JSB G^EXESMAXACMODE :PRIVILEGE
 55 50 D0 0349 1330 60\$: MOVL R0,R5 :MAXIMIZED WITH MODE OF CALLER TO
 034C 1331 :DETERMINE ACCESS MODE OF NEW TABLE
 57 04 AC D0 034C 1332 : MOVL CNATTR(AP),R7 :TABLE ATTRIBUTES
 12 13 0350 1333 : BEQL 200\$:BRANCH IF NOT PRESENT
 57 57 67 D0 0352 1334 : IFNORD #4,(R7),9000\$:CHECK READ ACCESS
 FFFFFFF8 8F D3 0358 1335 : MOVL (R7),R7 :GET VALUE
 035B 1336 : BITL #^C<-
 0362 1337 : LNMSM_CONFINE! - :CONFINE TO PROCESS
 0362 1338 : LNMSM_CRELOG! - :\$CRELOG USED TO CREATE NAME
 0362 1339 : LNMSM_NO_ALIAS - :DO NOT ALLOW ALIASES
 BB 12 0362 1340 :>R7 :INVALID TABLE ATTRIBUTES
 0364 1341 : BNEQ 9010\$:
 5A 0C AC D0 0364 1343 200\$: MOVL CNLOGNAM(AP),R10 :ADDRESS OF LOGICAL NAME DESCRIPTOR
 B5 13 0368 1344 : BEQL 9010\$:ERROR IF NOT PRESENT
 036A 1345 : IFNORD #8,(R10),9000\$:CHECK ACCESS TO DESCRIPTOR
 50 6A 7E 0370 1346 : MOVAQ (R10),R0 :ADDRESS OF DESCRIPTOR
 FC8A' 30 0373 1347 : BSBW LNMSPROBER :PROBE LOGICAL NAME STRING
 AF 50 E9 0376 1348 : BLBC R0,9030\$:BRANCH IF CAN'T READ OR NOT PRESENT
 7E 51 7D 0379 1349 : MOVQ R1,-(SP) :SAVE LOGICAL NAME DESCRIPTOR
 5A 5E D0 037C 1350 : MOVL SP,R10 :ADDRESS OF LOGICAL NAME DESCRIPTOR
 037F 1351 :
 59 08 AC D0 037F 1352 : MOVL CNTABNAM(AP),R9 :ADDRESS OF TABLE NAME DESCRIPTOR
 9A 13 0383 1353 : BEQL 9010\$:ERROR IF NOT PRESENT
 0385 1354 : IFNORD #8,(R9),9000\$:CHECK ACCESS TO DESCRIPTOR
 50 69 7E 038B 1355 : MOVAQ (R9),R0 :SET UP CALL
 FC6F' 30 038E 1356 : BSBW LNMSPROBER :PROBE TABLE NAME STRING
 94 50 E9 0391 1357 : BLBC R0,9030\$:BRANCH IF CAN'T READ OR NOT PRESENT
 7E 51 7D 0394 1358 : MOVQ R1,-(SP) :SAVE TABLE NAME DESCRIPTOR
 59 5E D0 0397 1359 : MOVL SP,R9 :ADDRESS OF TABLE NAME DESCRIPTOR
 039A 1360 :
 039A 1361 : PROBE AND VERIFY THE ITEM LIST. THIS IS THE FIRST OF TWO PASSES DOWN
 039A 1362 : THE ITEM LIST. THIS FIRST PASS IS BEING DONE TO ACCUMULATE
 039A 1363 : INFORMATION AS TO THE NUMBER OF TRANSLATION STRINGS BEING DEFINED AND
 039A 1364 : THEIR COLLECTIVE SIZE FOR USE IN DETERMINING THE AMOUNT OF SPACE TO
 039A 1365 : ALLOCATE FOR THE NEW LOGICAL NAME TABLE ENTRY. VALID ITEM TYPES FOR
 039A 1366 : THIS SYSTEM SERVICE ARE:
 039A 1367 :
 039A 1368 : ATTRIBUTES - ATTRIBUTES FOR NEXT TRANSLATION STRING
 039A 1369 : INDEX - INDEX TO BE ASSIGNED TO NEXT TRANSLATION STRING
 039A 1370 : STRING - EQUIVALENCE STRING
 039A 1371 : TABLE - LOGICAL NAME TABLE NAME (OUTPUT ITEM)
 039A 1372 : LNMBADDR - LOGICAL NAME LNMB ADDRESS (OUTPUT ITEM)
 039A 1373 :
 039A 1374 :
 039A 1375 :
 039A 1376 : CLRQ -(SP) :CLEAR COUNTERS

5B 14 AC D0 039C 1377 MOVL CNITMLST(AP),R11 ;ADDRESS OF ITEM LIST
 SD 13 03A0 1378 BEQL 2010\$;BRANCH IF NONE
 50 50 D4 03A2 1379 CLRL R0 ;CHAINED ITEM LIST COUNTER
 58 5B D0 03A4 1380 MOVL R11,R8 ;USER R8 TO VERIFY ITEM LIST
 03A7 1381 1005\$: IFNORD #4,(R8),1021\$;CHECK IF FIRST LONGWORD READABLE
 03AD 1382 1010\$:
 52 02 A8 3C 03AD 1383 MOVZWL 2(R8),R2 ;GET ITEM TYPE
 4C 13 03B1 1384 BEQL 2010\$;DONE IF ITEM TYPE IS ZERO
 03B3 1385 IFNORD #12,4(R8),1021\$;CHECK REST OF THIS DESCRIPTOR
 03BA 1386
 03BA 1387
 03BA 1388 CASE R2,<- ;PLUS FIRST LONGWORD OF NEXT ONE
 03BA 1389 1040\$, - ;HANDLE EACH ITEM TYPE SEPARATELY
 03BA 1390 1030\$, - ;INDEX ITEM
 03BA 1391 1040\$, - ;STRING ITEM
 03BA 1392 1040\$, - ;ATTRIBUTES ITEM
 03BA 1393 >,W,#1 ;TABLE ITEM
 52 09 B1 03C6 1394 CMPW #LNMS_LNMB_ADDR, R2 ;TEST FOR LNMB_ADDR
 1D 13 03C9 1395 BEQL 1040\$
 52 FFFF 8F B1 03CB 1396 CMPW #LNMS_CHAIN,R2 ;TEST FOR CHAINED LIST
 1C 13 03D0 1397 BEQL 1050\$
 FF4A 31 03D2 1398 1015\$: BRW 9010\$;ILLEGAL ITEM TYPE
 04 03D5 1399 1020\$: RET ;RETURN ANY ERRORS
 FF42 31 03D6 1400 1021\$: BRW 9000\$;ACCESS VIOLATION
 03D9 1402
 03D9 1403 ; STRING ITEM.
 03D9 1404
 03D9 1405
 03D9 1406 ; VALIDATE THE ITEM. IF THE STRING SPECIFIED IS A VALID EQUIVALENCE
 03D9 1407 ; STRING THEN INCREMENT THE TRANSLATION STRING COUNTER, AND ADD THE
 03D9 1408 ; SIZE OF THIS STRING TO THE TOTAL STRING SIZE BEING ACCUMULATED.
 03D9 1409
 03D9 1410
 50 68 7E 03D9 1411 1030\$: MOVAQ (R8),R0 ;ADDRESS OF STRING ITEM DESCRIPTOR
 FC21. 30 03DC 1412 BSBW LNMS\$PROBER ;PROBE TRANSLATION STRING
 F3 50 E9 03DF 1413 BLBC R0,1020\$;BRANCH IF CAN'T READ OR NOT PRESENT
 6E 6E D6 03E2 1414 INCL (SP) ;INCREMENT EQUIVALENCE STRING COUNTER
 04 AE 51 C0 03E4 1415 ADDL2 R1,4(SP) ;ADD SIZE OF STRING TO TOTAL SIZE
 03E8 1416
 03E8 1417 ; POSITION TO THE NEXT ITEM IN THE ITEM LIST.
 03E8 1418
 03E8 1419 ;
 03E8 1420
 58 0C CO 03E8 1421 1040\$: ADDL2 #12,R8 ;POSITION TO NEXT ITEM
 FFBF 31 03EB 1422 BRW 1010\$;AND CONTINUE ITEM LIST VERIFICATION
 03EE 1423
 03EE 1424
 03EE 1425 ; POSITION TO THE NEXT CHAIN OF THE CHAINED ITEM LIST.
 03EE 1426
 03EE 1427
 0400 8F 50 D6 03EE 1428 1050\$: INCL R0 ;COUNT ONE MORE
 50 B1 03F0 1429 CMPW R0,#1024 ;TOO MANY - ASSUME A LOOP
 DB 14 03F5 1430 BGTR 1015\$;GET POINTER TO NEXT LIST
 58 04 A8 D0 03F7 1431 MOVL 4(R8),R8 ;LAST ITEM
 02 13 03FB 1432 BEQL 2010\$;PROCESS THIS ONE
 A8 11 03FD 1433 BRB 1005\$

03FF 1434
 03FF 1435
 03FF 1436 : RAISE IPL TO AST DELIVERY LEVEL SO THAT THERE ARE NO
 03FF 1437 : INTERRUPTIONS WHILE THE NEW LOGICAL NAME ENTRY IS BEING CREATED.
 03FF 1438 :
 03FF 1439 :
 03FF 1440 2010\$: SETIPL S^#IPLS_ASTDEL ;RAISE TO AST DELIVERY LEVEL
 0402 1441
 0402 1442 :
 0402 1443 : ALLOCATE A LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME TABLE ENTRY.
 0402 1444 : ASSUME THAT THE NEW LOGICAL NAME TABLE WILL BE SHAREABLE, SO ALLOCATE
 0402 1445 : THE SPACE FROM SYSTEM PAGED POOL. UNTIL THE LOGICAL NAME TABLE IN
 0402 1446 : WHICH THE NEW ENTRY WILL BE LOCATED IS FOUND, WHETHER THE NEW
 0402 1447 : LOGICAL NAME TABLE ENTRY WILL BE IN SYSTEM POOL AND SHAREABLE OR IN
 0402 1448 : P1 SPACE AND NOT SHAREABLE IS NOT KNOWN. HOWEVER, THE LOGICAL NAME
 0402 1449 : MUTEX MUST BE LOCKED BEFORE A SEARCH CAN BE MADE FOR THE LOGICAL
 0402 1450 : NAME TABLE, AND ONCE THE MUTEX IS LOCKED, SYSTEM POOL CAN NOT
 0402 1451 : ALTHOUGH P1 POOL CAN BE ALLOCATED. THEREFORE, THE NEW LOGICAL NAME
 0402 1452 : TABLE ENTRY IS ASSUMED TO BE SHAREABLE, AND IF THE LOGICAL NAME
 0402 1453 : TABLE WHICH TO CONTAIN THE NEW ENTRY IS FOUND TO RESIDE IN
 0402 1454 : SYSTEM POOL, THEN THE SYSTEM POOL ALLOCATED FOR THE NEW ENTRY IS
 0402 1455 : DEALLOCATED, AND PROCESS-PRIVATE POOL IS ALLOCATED IN ITS PLACE.
 0402 1456 :
 0402 1457
 51 6E 04 C5 0402 1458 MULL3 #LNMXST_XLATION,(SP),R1 :LNMXS OVERHEAD +
 51 8E C0 0406 1459 ADDL2 (SP)+,RT :TRANSLATION STRING COUNT FIELDS +
 51 8E C0 0409 1460 ADDL2 (SP)+,R1 :TRANSLATION STRINGS SIZES +
 51 6A C0 040C 1461 ADDL2 (R10),R1 :LOGICAL NAME SIZE +
 CO 040F 1462 ADDL2 #<- :LOGICAL NAME COUNT FIELD +
 0410 1463 LNMB\$T_NAME+1+ - :LNMB OVERHEAD +
 51 13 0410 1464 1 - :TRAILER BYTE =
 0410 1465 >,R1 :SIZE OF SYSTEM POOL TO ALLOCATE
 0412 1466
 00000000'GF 03 50 16 0412 1467 JSB G^EXE\$ALOPAGED :ALLOCATE THE LOGICAL NAME TABLE ENTRY
 FF05 31 E8 0418 1468 BLBS R0,2040\$:BRANCH IF SUFFICIENT MEMORY
 08 A2 51 B0 041B 1469 BRW 9020\$:OTHERWISE RETURN ERROR
 041E 1470 2040\$: MOVW R1,LNMB\$W_SIZE(R2) :SAVE NEW BLOCK SIZE WITHIN THE BLOCK
 0422 1471
 0422 1472 :
 0422 1473 : WRITE LOCK THE LOGICAL NAME MUTEX AND THEN POSITION TO THE LOGICAL NAME
 0422 1474 : TABLE THAT IS TO CONTAIN THE NEW LOGICAL NAME TABLE ENTRY. DEALLOCATE THE NEW
 0422 1475 : LNMB IF UNABLE TO POSITION TO ANY TABLE ENTRY.
 0422 1476 :
 0422 1477 :
 14 BB 0422 1478 PUSHR #^M<R2,R4> :SAVE ADDRESS OF PCB AND NEW LNMB
 FB09 30 0424 1479 BSBW LNMSLOCKW :LOCK TABLES FOR WRITING
 51 55 D0 0427 1480 MOVL R5,R1 :ACCESS MODE
 52 69 7D 042A 1481 MOVQ (R9),R2 :TABLE NAME DESCRIPTOR
 FB00 30 042D 1482 BSBW LNMSFIRSTTAB :SEARCH FOR SPECIFIED TABLE
 54 BED0 0430 1483 POPL R4 :RESTORE ADDRESS OF NEW LNMB
 50 DD 0433 1484 PUSHL R0 :SAVE RETURN STATUS
 03 50 E8 0435 1485 BLBS R0,2050\$:BRANCH ON FAILURE
 021B 31 0438 1486 BRW 4020\$
 0438 1487
 0438 1488 :
 0438 1489 : IF THE NEW LOGICAL NAME TABLE ENTRY WILL BE SHAREABLE BECAUSE THE LOGICAL
 0438 1490 : NAME TABLE WHICH WILL CONTAIN THE NEW ENTRY HAS BEEN FOUND TO BE, THEN MAKE

```

        043B 1491 : SURE THE NEW LOGICAL NAME TABLE ENTRY WILL NOT BE MARKED AS CONFINED AND
        043B 1492 : MAKE SURE THE CALLER HAS WRITE ACCESS TO THE CONTAINING TABLE.
        043B 1493
        1B 00 E1 043B 1494 2050$: BBC #LNMTHSV SHAREABLE,- :BRANCH IF THE CONTAINING TABLE ENTRY
      57 02 CA 043D 1495 LNMTHSB FLAGS(R1),2070$ :IS IN PROCESS-PRIVATE SPACE
        0442 1496 BICL2 #LNMSM_CONFINE,R7 :SHAREABLE NAMES CAN'T BE CONFINED
        52 02 9A 0442 1497 MOVZBL #WRITE_ACCESS,R2 :CODE FOR ACCESS CHECK
        54 54 DD 0445 1499 PUSHL R4 :SAVE LNMB PTR
        FBB2' 30 044B 1500 MOVL 8(SP),R4 :RESTORE PCB
        54 8E DO 044E 1501 BSBW LNMSCHECK_PROT :CHECK THE PROTECTION
        41 50 E8 0451 1502 MOVL (SP)+R4 :RESTORE LNMB
        6E 50 DO 0454 1503 BLBS R0,2090$ :OK
        01FC 31 0457 1504 MOVL R0,(SP) :SAVE STATUS
        045A 1505 BRW 4020$ :AND EXIT

        045A 1506
        045A 1507 :
        045A 1508 : IF THE LOGICAL NAME TABLE THAT IS TO CONTAIN THE NEW LOGICAL NAME TABLE ENTRY
        045A 1509 : IS DISCOVERED TO EXIST IN PROCESS-PRIVATE SPACE, THEN THIS IS ALSO WHERE THE
        045A 1510 : NEW LOGICAL NAME ENTRY SHOULD GO. DEALLOCATE THE SYSTEM SPACE LNMB, AND
        045A 1511 : ALLOCATE THE SAME SIZE BLOCK FROM P1 SPACE.
        045A 1512 :
        045A 1513 2070$: PUSHL R1 :SAVE ADDRESS OF CONTAINING TABLE HEADER
        7E 08 A4 3C 045C 1514 MOVZWL LNMB$W_SIZE(R4),-(SP) :SAVE SIZE OF LNMB
        50 54 DO 0460 1515 MOVL R4,R0 :ADDRESS OF SYSTEM SPACE LNMB
        FB9A' 30 0463 1516 BSBW LNMSDELBLK :DEALLOCATE SYSTEM SPACE LNMB
        045A 1517
        045A 1518
        0466 1519 POPL R1 :SIZE OF STORAGE TO ALLOCATE
        00000000'GF 16 0469 1520 JSB G^EXE$ALOP1PROC :ALLOCATE NEW LNMB FROM P1 SPACE
        OC 50 E8 046F 1521 BLBS R0,2080$ :BRANCH IF SUCCESSFUL
        04 AE 0124 8F 3C 0472 1522 MOVZWL #SS$ INSFMEM,4(SP) :OTHERWISE RETURN INSUFFICIENT MEMORY
        5E 04 C0 0478 1523 ADDL2 #4,SP :THROW AWAY TABLE HEADER ADDRESS
        01DE 31 047B 1524 BRW 4030$ :RETURN ERROR AFTER RELEASING MUTEX

        045A 1525
        08 A4 52 DO 047E 1526 2080$: MOVL R2,R4 :ADDRESS OF NEW LNMB
        51 8E DO 0481 1527 MOVW R1,LNMB$W_SIZE(R4) :SAVE NEW BLOCKS SIZE WITHIN BLOCK
        52 09 A1 8E 0485 1528 POPL R1 :ADDRESS OF CONTAINING TABLE HEADER
        01 E1 0488 1529 MOVL LNMTHSL_NAME(R1),R2 :ADDRESS OF TABLE'S LNMB
        04 10 A2 048E 1530 BBC #LNMSV_CONFINE- :IS THE ENTRY MARKED CONFINE?
        00 57 01 E3 0491 1531 LNMBSB_FLAGS(R2),2090$ :BRANCH IF NOT
        0495 1532 BBCS #LNMSV_CONFINE,R7,2090$ :MAKE SURE NEW ENTRY IS IF SO
        0495 1533
        0495 1534 :
        0495 1535 : MAKE SURE THAT THE QUOTA HOLDER OF THE LOGICAL NAME TABLE WHICH IS TO CONTAIN
        0495 1536 : THE NEW LOGICAL NAME TABLE ENTRY HAS SUFFICIENT QUOTA FOR THE CREATION OF
        0495 1537 : THIS NEW LOGICAL NAME TABLE ENTRY.
        0495 1538 :
        0495 1539
        52 19 A1 DO 0495 1540 2090$: MOVL LNMTHSL_QTABLE(R1),R2 :RETRIEVE QUOTA HOLDER'S ADDRESS
        50 08 A4 3C 0499 1541 MOVZWL LNMB$W_SIZE(R4),R0 :SIZE OF QUOTA TO BE WITHDRAWN
        049D 1542
        21 A2 50 D1 049D 1543 CMPL R0,LNMTHSL_BYTES(R2) :IS THEIR SUFFICIENT QUOTA?
        08 1B 04A1 1544 BLEQU 3000$ :YES - THEN CONTINUE
        6E 224C 8F 3C 04A3 1545 MOVZWL #SS$ EXLNMQOUTA,(SP) :NO - THEN DEALLOCATE LNMB AND RETURN
        01AB 31 04AB 1546 BRW 4020$ :EXCEEDED QUOTA ERROR
        04AB 1547

```

04AB 1548 : BUILD THE LOGICAL NAME TABLE ENTRY. REGISTER USAGE IS AS FOLLOWS:

04AB 1549 : R1 = ADDRESS OF THE LOGICAL NAME TABLE'S TABLE HEADER.

04AB 1550 : R4 = ADDRESS OF LOGICAL NAME BLOCK FOR NEW LOGICAL NAME TABLE ENTRY.

04AB 1551 : R5 = ACCESS MODE.

04AB 1552 : R7 = ATTRIBUTE BITS.

04AB 1553 : R9 = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.

04AB 1554 : R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.

04AB 1555 : R11 = ADDRESS OF ITEM LIST.

04AB 1556 :
 50 0A A4 9E 04AB 1560 3000\$: MOVAB LNMB\$B_TYPE(R4),R0 ;POSITION TO BLOCK TYPE
 80 40 8F 90 04AF 1561 MOVB #DYN\$C_LNM,(R0)+ ;SET DATA STRUCTURE TYPE
 80 55 90 04B3 1562 MOVB R5,(R0)+ ;SET OWNER ACCESS MODE
 80 51 00 04B6 1563 MOVL R1,(R0)+ ;ADDRESS OF CONTAINING TABLE HEADER
 80 57 90 04B9 1564 MOVB R7,(R0)+ ;STORE NAME ATTRIBUTES
 80 6A 90 04BC 1565 MOVB (R10),(R0)+ ;STORE LENGTH OF NAME
 54 DD 04BF 1566 PUSHL R4 ;SAVE LNMB ADDRESS ON STACK
 01 E0 04C1 1567 BBS #LNMTHSV DIRECTORY,- ;BRANCH IF THIS LOGICAL NAME TO BE
 60 04 BA 07 61 04C3 1568 LNMTHSB_FLAGS(R1),3005\$;CONTAINED WITHIN A DIRECTORY HEADER
 60 04 BA 6A 28 04C5 1569 MOVC3 (R10),#4(R10),(R0) ;MOVE LOGICAL NAME INTO LNMB NAME FIELD
 21 11 04CA 1570 BRB 3010\$;JOIN COMMON CODE
 04CC 1571 :
 1F 6A B1 04CC 1572 3005\$: CMPW (R10),#LNMSC_TABNAMLEN ;SIZE EXCEED MAX FOR NAME IN DIRECTORY?
 08 1B 04CF 1573 BLEQU 3009\$;CONTINUE IF IT DOESNT
 6E 0154 54 8F BED0 04D1 1574 3007\$: POPL R4 ;OTHERWISE RESTORE LNMB ADDRESS
 017A 3C 04D4 1575 MOVZWL #SSS_IVLOGNAM,(SP) ;SET ERROR CODE
 017A 31 04D9 1576 BRW 4020\$;GO RETURN ERROR
 04DC 1577 :
 00 04 BA 6A 2F 04DC 1578 3009\$: MOVTUC (R10),#4(R10),#0,- ;CHECK FOR PROPER LOGICAL NAME SYNTAX
 00000000'EF 04E1 1579 EXE\$LN - SYNTAX_DAT,- ;AND AT THE SAME TIME MOVE THE LOGICAL
 60 6A 04E6 1580 (R10),(R0) ;NAME INTO THE LNMB NAME FIELD
 53 E7 1D 04E8 1581 BVS 3007\$;GO RETURN ERROR IF IMPROPER SYNTAX
 53 55 00 04EA 1582 MOVL R5,R3 ;ELSE SHIFT LNMX ADDR INTO PROPER REG
 54 BED0 04ED 1583 3010\$: POPL R4 ;RESTORE LNMB ADDRESS FROM STACK
 04F0 1584 :
 04F0 1585 : FILL IN THE LNMX PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME
 04F0 1586 : TABLE ENTRY. THIS INVOLVES PROCESSING THE ITEM LIST FOR A SECOND TIME SO
 04F0 1587 : THAT THE LOGICAL NAME TRANSLATION BLOCKS MAYBE CONSTRUCTED, AND ANY OUTPUT
 04F0 1588 : INFORMATION RETURNED.
 04F0 1589 :
 04F0 1590 :
 04F0 1591 :
 5B D5 04F0 1592 TSTL R11 ;WAS AN ITEM LIST DEFINED?
 03 12 04F2 1593 BNEQ 3030\$;YES - GO PROCESS IT
 013C 31 04F4 1594 3020\$: BRW 3950\$;NO - SKIP ITEM LIST PROCESSING
 04F7 1595 :
 56 D4 04F7 1596 3030\$: CLRL R6 ;0 IS THE DEFAULT INDEX
 58 D4 04F9 1597 CLRL R8 ;NO ATTRIBUTES DEFINED AS YET
 04FB 1598 3035\$: IFNORD #4,(R11),3060\$;CHECK IF FIRST LONGWORD READABLE
 0501 1599 3040\$:
 52 02 AB 3C 0501 1600 MOVZWL 2(R11),R2 ;GET ITEM TYPE
 ED 13 0505 1601 BEQL 3020\$;DONE IF ITEM TYPE IS ZERO
 0507 1602 IFNORD #12,4(R11),3060\$;CHECK REST OF THIS DESCRIPTOR
 050E 1603 : PLUS FIRST LONGWORD OF NEXT ONE
 050E 1604 :

			050E	1605	CASE	R2,<-		HANDLE EACH ITEM TYPE SEPARATELY	
			050E	1606			3200\$, -	INDEX ITEM	
			050E	1607			3300\$, -	STRING ITEM	
			050E	1608			3200\$, -	ATTRIBUTES ITEM	
			050E	1609			3400\$, -	TABLE ITEM	
			050E	1610		>,W,#1			
52	09	B1	051A	1611	CMPW	#LNMS_LNMB_ADDR, R2		;TEST FOR SPECIAL ITEM	
	15	13	051D	1612	BEQL	3100\$			
52	FFFF	8F	81	051F	CMPW	#LNMS_CHAIN,R2		;TEST FOR CHAINED ITEM LIST	
	06	13	0524	1614	BEQL	3070\$			
	0100	31	0526	1615	3050\$:	BRW	3930\$;ILLEGAL ITEM	
	00F8	31	0529	1616	3060\$:	BRW	3920\$;REPORT ACCESS VIOLATION	
			052C	1617					
			052C	1618				: POSITION TO THE NEXT CHAIN IN THE CHAINED ITEM LIST.	
			052C	1619					
			052C	1620					
			052C	1621					
5B	04	AB	D0	052C	1622	3070\$:	MOVL	4(R11),R11	:GET POINTER
	C2	13	0530	1623	BEQL	3020\$:LAST ITEM	
	C7	11	0532	1624	BRB	3035\$:GO PROCESS IT	
			0534	1625					
			0534	1626					
			0534	1627				:LOGICAL NAME BLOCK ADDRESS	
			0534	1628					
			0534	1629				:TEST IF THE LOGICAL NAME BLOCK ADDRESS CAN BE WRITTEN IN THE	
			0534	1630				:REQUESTED PLACE. (CURRENT USE IS TO LINK A LOGICAL NAME BLOCK	
			0534	1631				:WITH MTL OR MAILBOX UCB)	
			0534	1632					
			0534	1633					
6B	04	B1	0534	1634	3100\$:	CMPW	#4,(R11)	:IS THE ITEM A LONGWORD	
50	04	AB	D0	0537	BNEQ	3050\$:ERROR IF NOT	
	ED	12	0537	1635	MOVL	4(R11),R0		:RETRIEVE ITEM BUFFER ADDRESS	
			0539	1636	IFNOWRT	#4,(R0),3060\$:ACCVIO IF ITEM BUFFER NOT WRITEABLE	
60	54	D0	0543	1637	MOVL	R4,(R0)		:MOVE THE ADDRESS OF LNM TO BUFFER	
	00D5	31	0546	1638	BRW	3910\$:GO POSITION TO NEXT ITEM	
			0549	1639					
			0549	1640					
			0549	1641					
			0549	1642				:INDEX OR ATTRIBUTES ITEM.	
			0549	1643					
			0549	1644				:VALIDATE THE ITEM. SAVE THE INDEX OR ATTRIBUTES FOR APPLICATION TO	
			0549	1645				:THE NEXT TRANSLATION BLOCK THAT IS TO BE CREATED.	
			0549	1646					
			0549	1647					
6B	04	B1	0549	1648	3200\$:	CMPW	#4,(R11)	:IS ITEM BUFFER AT LEAST A LONGWORD?	
50	04	AB	D0	054C	BGTRU	3050\$:ERROR IF NOT	
	D8	1A	054C	1649	MOVL	4(R11),R0		:RETRIEVE ITEM BUFFER ADDRESS	
			054E	1650	IFNORD	(R11),(R0),3060\$:ACCVIO IF ITEM BUFFER NOT READABLE	
50	60	D0	0552	1651	MOVL	(R0),R0		:PICK UP THE LONGWORD	
			0558	1652					
			055B	1653					
52	01	B1	055B	1654	CMPW	#LNMS_INDEX,R2		:IS IT AN INDEX ITEM?	
	11	13	055E	1655	BEQL	3210\$:YES - BRANCH AND VERIFY INDEX ITEM	
50	FFFFFCFF	8F	D3	0560	BITL	#^C< -		:TEST FOR INVALID ATTRIBUTES	
			0567	1656				:TRANSLATION IS CONCEALED	
			0567	1657				:TRANSLATION IS TERMINAL	
			0567	1658					
			0567	1659					
58	50	F8	BD	12	0567	1660	BNEQ	3050\$:INVALID ATTRIBUTES ITEM
			9C	0569	1661		ROT1	#-8,R0,R8	:SAVE ATTRIBUTES FOR LNMX CREATION

00AD 31 056E 1662 BRW 3910\$:GO POSITION TO NEXT ITEM
50 FFFFFF80 8F D3 0571 1663 3210\$: BITL #^C127, R0 :IS INDEX BETWEEN 0 AND 127?
16 13 0578 1664 BEQL 3220\$:YES - NO PRIVILEGE NEEDED
057A 1665
057A 1666
057A 1667
057A 1668 : THIS IS HERE SO THAT MTL OR MAILBOX UCB CAN HAVE ITS ADDRESS
057A 1669 : STORED IN THE LOGICAL NAME BLOCK.
057A 1670
057A 1671
057A 1672
52 00C00000 52 DC 057A 1673 MOVPSL R2 :GET THE PSL
8F D3 057C 1674 BITL #PSL\$M_PRVMOD, R2 :TEST IF PREVIOUS MODE WAS KERNEL
A1 12 0583 1675 BNEQ 3050\$:NO - BAD INDEX VALUE
50 FFFFFF81 8F D1 0585 1676 CMPL #LNMXSC_BACKPTR, R0 :CORRECT INDEX
98 12 058C 1677 BNEQ 3050\$:NO - BAD INDEX VALUE
05 11 058E 1678 BRB 3225\$
56 50 D1 0590 1679 3220\$: CMPL R0, R6 :IS INDEX LESS THAN LAST ONE SEEN?
91 19 0593 1680 BLSS 3050\$:YES - INVALID INDEX ITEM
56 50 D0 0595 1681 3225\$: MOVL R0, R6 :SAVE INDEX FOR LNMX CREATION
0083 31 0598 1682 BRW 3910\$:GO POSITION TO NEXT ITEM
059B 1683
059B 1684 : STRING ITEM.
059B 1685
059B 1686
059B 1687 : VALIDATE THE ITEM, INCLUDING THAT THE STRING SPECIFIED IS A VALID EQUIVALENCE
059B 1688 : STRING. VERIFY THAT THE CREATION OF THIS TRANSLATION BLOCK WILL NOT EXCEED
059B 1689 : THE AMOUNT OF SPACE ALLOCATED. FILL IN THE NEXT TRANSLATION BLOCK UTILIZING
059B 1690 : THE STRING SPECIFIED AS THE TRANSLATION STRING AS WELL AS THE CURRENT INDEX
059B 1691 : AND ATTRIBUTES SETTINGS. INCREMENT THE CURRENT INDEX.
059B 1692 :
059B 1693
50 6B 7E 059B 1694 3300\$: MOVAQ (R11), R0 :ADDRESS OF STRING ITEM DESCRIPTOR
FA5F 30 059E 1695 BSBW LNMSPROBER :PROBE TRANSLATION STRING
42 50 E9 05A1 1696 BLBC R0, 3410\$:BRANCH IF CAN'T READ OR NOT PRESENT
50 53 54 C3 05A4 1698 SUBL3 R4, R3, R0 :NUMBER OF BYTES CURRENTLY IN USE
50 06 C0 05A8 1699 ADDL2 #LNMX\$T_XLATION+2, R0 :ADD OVERHEAD FOR CURRENT TRANSLATION
50 51 C0 05AB 1700 ADDL2 R1, R0 :ADD IN SIZE OF CURRENT TRANSLATION
08 A4 50 B1 05AE 1701 CMPW R0, LNMB\$W_SIZE(R4) :EXCEED LOGICAL NAME BLOCK SIZE?
75 1A 05B2 1702 BGTRU 3930\$:YES - RETURN AN ERROR
05B4 1703
83 58 90 05B4 1704 MOVB R8, (R3)+ :CURRENT ATTRIBUTES BECOME LNMX FLAGS
83 56 90 05B7 1705 MOVB R6, (R3)+ :CURRENT INDEX LEVEL BECOMES LNMX LEVEL
83 B4 05BA 1706 CLRW (R3)+ :INITIALIZE HASH CODE LOCATION
83 56 D6 05BC 1707 INCL R6 :INCREMENT INDEX LEVEL BY 1
83 51 90 05BE 1708 MOVB R1, (R3)+ :STORE SIZE OF TRANSLATION STRING
54 DD 05C1 1709 PUSHL R4 :SAVE NEEDED REGISTERS OVER MOVC3
63 62 51 28 05C3 1710 MOVC3 R1, (R2), (R3) :MOVE TRANSLATION STRING INTO LNMX
54 8ED0 05C7 1711 POPL R4 :RESTORE REGISTERS
52 11 05CA 1712 BRB 3910\$:GO POSITION TO NEXT ITEM
05CC 1713
05CC 1714 : TABLE ITEM.
05CC 1715
05CC 1716
05CC 1717 : VALIDATE THE ITEM. STORE THE NAME OF THE LOGICAL NAME TABLE WHICH IS TO
05CC 1718 : CONTAIN THIS NEW LOGICAL NAME TABLE ENTRY IN THE ITEM BUFFER PROVIDED, AND

05CC 1719 : THE SIZE OF THE NAME STORED IN THE ITEM SIZE BUFFER, IF PROVIDED.
 05CC 1720 :
 05CC 1721 :

50 7E 51 04	6B 6E 3C 05CF	7D 05D2 05D6 05D8	05CC 1722 1723 1724 1725 1726 1727 1728 1729 1730	3400\$: MOVQ (R11),-(SP) MOVZWL (SP),R1 MOVL 4(SP),R0 PUSHL R3 CLRL R3 JSB G^EXE\$PROBEW POPL R3 MOVQ (SP)+,R1 BLBC R0,3940\$	SAVE DESCRIPTOR OF TABLE ITEM BUFFER SIZE OF TABLE ITEM BUFFER ADDRESS OF TABLE ITEM BUFFER SAVE CURRENT LNMX POSITION ACCESS MODE PROBE TABLE ITEM BUFFER FOR WRITTING RESTORE CURRENT LNMX POSITION RESTORE TABLE ITEM BUFFER DESCRIPTOR RETURN IF NOT WRITEABLE
00000000'GF	16 53 8ED0	05DA 05E0 05E3 05E6	1723 1724 1725 1726 1727 1728 1729 1730 1731	MOVL LNMB\$L_TABLE(R4),R5 MOVL LNMB\$H_NAME(R5),R5 MOVAB LNMB\$T_NAME(R5),R5 MOVZBL (R5)+,R0	ADDRESS OF CONTAINING TABLE HEADER ADDRESS OF CONTAINING LNMB ADDRESS OF COUNT FIELD SIZE OF CONTAINING TABLE'S NAME
55 51	0C A4 09 A5 11 A5	D0 05ED 05F1 05F5 05F8	1732 1733 1734 1735 1736	CMPW R0,R1 BLEQ 3420\$	ENOUGH ROOM IN BUFFER FOR NAME? BRANCH IF SUFFICIENT ROOM
6E	0601 8F	3C 05FD	1738 1739	MOVZWL #SSS_BUFFEROVF,(SP)	CHANGE RETURN STATUS
	50 51	3C 0602	1740 1741	MOVZWL R1,R0	RETURN AS MUCH AS POSSIBLE
51	08 AB 09 13	D0 0605 0609	1742 1743	MOVL 8(R11),R1 BEQL 3430\$	WAS A RETURN LENGTH BUFFER SPECIFIED NO - DON'T RETURN LENGTH
	0611	0614	1744 1745	IFNOWRT #2,(R1),3920\$ MOVW R0,(R1)	YES - PROBE RETURN LENGTH BUFFER AND RETURN LENGTH
62	7E 53	7D 0614	1747 1748	3430\$: MOVQ R3,-(SP) MOV C3 R0,(R5),(R2)	SAVE REGISTERS OVER MOVC3 RETURN CONTAINING TABLE NAME STRING
	65 53	28 0617	1749	MOVQ (SP)+,R3	RESTORE SAVED REGISTERS
5B	0C FEDD	C0 061E 31 0621	1750 1751 1752	ADDL2 #12,R11 BRW 3040\$	POSITION TO NEXT ITEM AND CONTINUE ITEM LIST VERIFICATION
6E	0C 2D	3C 0624 11 0627	1753 1754 1755	MOVZWL #SSS_ACCVIO,(SP) BRB 4020\$	ACCESS VIOLATION DEALLOCATE NEW LNMB AND RETURN ERROR
6E	14 28	3C 0629 11 062C	1756 1757 1758	MOVZWL #SSS_BADPARAM,(SP) BRB 4020\$	BAD PARAMETER SEEN DEALLOCATE NEW LNMB AND RETURN ERROR
6E	50 23	D0 062E 11 0631	1759 1760 1761	MOVL R0,(SP) BRB 4020\$	SAVE RETURN STATUS DEALLOCATE NEW LNMB AND RETURN ERROR
	0633	0633	1762 1763	:	
	0633	0633	1764 1765 1766	: FILL IN THE LAST LNMX PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME TABLE ENTRY. THE LAST TRANSLATION BLOCK CONSISTS SOLELY OF A FLAGS FIELD.	
	0633	0633	1767 1768	: AND IS MARKED WITHIN THIS FLAGS FIELD AS THE LAST LNMX.	
83	04	90 0633	1769 1770	3950\$: MOVB #LNMXSM_XEND,(R3)+	;STORE END FLAG
	0636	0636	1771	:	
	0636	0636	1772 1773	: AT THIS POINT ALL CHECKS HAVE BEEN MADE, AND THE TABLE STRUCTURES MAY NOW BE MODIFIED FOR THE FIRST TIME. INSERT THE NEW LOGICAL NAME TABLE ENTRY.	
	0636	0636	1774 1775	: THIS INSERTION MAY TAKE ONE OF TWO FORMS:	

0636 1776					
0636 1777					1.) THE LOGICAL NAME TABLE ENTRY MAYBE INSERTED AS A NEW LOGICAL NAME. THE FOLLOWING CONDITIONS MUST HOLD TRUE.
0636 1778					A.) THERE IS NO LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AND ACCESS MODE WITHIN THE SAME CONTAINING LOGICAL NAME TABLE.
0636 1779					B.) THERE IS NO LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AND AN INNER ACCESS MODE WITHIN THE SAME CONTAINING LOGICAL NAME TABLE THAT DOES NOT ALLOW ALIASES.
0636 1780					
0636 1781					
0636 1782					
0636 1783					
0636 1784					
0636 1785					
0636 1786					
0636 1787					
0636 1788					
0636 1789					
0636 1790					
0636 1791					
0636 1792					
0636 1793					
0636 1794					
0636 1795					
51 54 D0 0636 1796			MOVL R4,R1		:ADDRES OF NEW LOGICAL NAME TABLE ENTRY
52 57 D0 0639 1797			MOVL R7,R2		:ATTRIBUTES
50 F9C1' 30 063C 1798			BSBW LNMSINSLOGTAB		:INSERT NEW LOGICAL NAME TABLE ENTRY
0631 8F B1 063F 1799			CMPW #SS\$ SUPERSEDE,RO		:INSERTION SUCCEED WITH SUPERSEDE?
0A 12 0644 1800			BNEQ 4010\$:BRANCH IF NO
6E 0601 8F B1 0646 1801			CMPW #SS\$ BUFFEROVF,(SP)		:BUFFER OVERFLOW SEEN?
OF 13 064B 1802			BEQL 4030\$:YES - THEN RETURN THAT STATUS
6E 50 D0 064D 1803			MOVL R0,(SP)		:OTHERWISE RETURN SUPERSEDE SUCCESS
09 50 E8 0650 1804	4010\$:		BLBS R0,4030\$:BRANCH ON SUCCESS
6E 50 D0 0653 1805			MOVL R0,(SP)		:OTHRWISE SAVE ERROR STATUS
0656 1806					
50 54 D0 0656 1807	4020\$:	MOVL R4,R0			:LOGICAL NAME BLOCK FOR NEW TABLE ENTRY
F9A4' 30 0659 1808		BSBW LNMSDELBLK			:DELETE BLOCK CONTAINING NEW TABLE ENTRY
065C 1809					
54 04 AE D0 065C 1810	4030\$:	MOVL 4(SP),R4			:RETRIEVE PCB ADDRESS
F99D' 30 0660 1811		BSBW LNMSUNLOCK			:UNLOCK TABLES
50 8ED0 0663 1812		POPL R0			:RESTORE STATUS
04 0666 1813		RET			:RETURN TO CALLER
0667 1814					
0667 1815		.DISABLE LSB			
0667 1816					
0667 1817 :		.PAGE			

0667 1819 .SBTTL EXESDELLNM - DELETE LOGICAL NAME
0667 1820 + EXESDELLNM - DELETE LOGICAL NAME
0667 1821 THIS SERVICE PROVIDES THE CAPABILITY TO:
0667 1822 1. DELETE A SPECIFIED LOGICAL NAME.
0667 1823 2. DELETE A SPECIFIED LOGICAL NAME TABLE.
0667 1824 3. DELETE ALL THE LOGICAL NAME TABLE ENTRIES CONTAINED WITHIN A
0667 1825 SPECIFIED LOGICAL NAME TABLE.
0667 1826
0667 1827
0667 1828
0667 1829
0667 1830 INPUTS:
0667 1831
0667 1832 DNACMODE(AP) = ADDRESS OF ACCESS MODE OF LOGICAL NAMES(S) TO BE DELETED.
0667 1833 DNLOGNAM(AP) = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR.
0667 1834 DNTABNAM(AP) = ADDRESS OF TABLE NAME STRING DESCRIPTOR.
0667 1835
0667 1836 R4 = CURRENT PROCESS PCB ADDRESS.
0667 1837
0667 1838 OUTPUTS:
0667 1839
0667 1840 RO LOW BIT CLEAR INDICATES FAILURE TO DELETE LOGICAL NAME(S).
0667 1841
0667 1842 RO = SSS_ACCVIO - TABLE NAME DESCRIPTOR, TABLE NAME STRING
0667 1843 LOGICAL NAME DESCRIPTOR, LOGICAL NAME STRING, CANNOT
0667 1844 BE READ BY CALLING ACCESS MODE.
0667 1845
0667 1846 RO = SSS_BADPARAM - INVALID ACCESS MODE SPECIFIED. NO TABLE
0667 1847 NAME DESCRIPTOR SPECIFIED.
0667 1848
0667 1849 RO = SSS_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
0667 1850 LOGICAL OR TABLE NAME STRING SPECIFIED.
0667 1851
0667 1852 RO = SSS_IVLOGTAB - INVALID TABLE NAME SPECIFIED.
0667 1853
0667 1854 RO = SSS_NOLOGNAM - LOGICAL NAME TABLE OR LOGICAL NAME
0667 1855 SPECIFIED DOES NOT EXIST.
0667 1856
0667 1857 RO = SSS_NOLOGTAB - LOGICAL NAME TABLE SPECIFIED DOES NOT EXIST.
0667 1858
0667 1859 RO = SSS_NOPRIV - PROCESS DOES NOT HAVE PRIVILEGE TO DELETE
0667 1860 THE SPECIFIED LOGICAL NAME(S).
0667 1861
0667 1862 RO = SSS_TOOMANYLN - TOO MANY LEVELS OF RECURSION IN SEARCH
0667 1863 FOR LOGICAL NAME TABLE OR LOGICAL NAME.
0667 1864
0667 1865 RO LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
0667 1866
0667 1867 RO = SSS_NORMAL - NORMAL COMPLETION. LOGICAL NAME(S) HAVE BEEN
0667 1868 DELETED.
0667 1869
0667 1870 SIDE EFFECTS:
0667 1871
0667 1872 THIS ROUTINE EXITS AT IPL 2.
0667 1873
0667 1874
0667 1875

```

0000000A 1876 .PSECT YSEXEPAGED
0FFC 000A 1877 .ENTRY EXESDELLNM,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0660' 31 000C 1878 BRW EXEDELLNM
000F 1879
00000667 1880 .PSECT YFSSLNM
0667 1881 .ENABLE LSB
50 0C 3C 0667 1882 900$: MOVZWL #SSS_ACCVIO,RO ;ACCESS VIOLATION
04 066A 1883 RET
50 14 3C 066B 1884 910$: MOVZWL #SSS_BADPARAM,RO ;BAD SYSTEM SERVICE PARAMETER
04 066E 1885 930$: RET
066F 1886
066F 1887 :
066F 1888 : VALIDATE AND COPY PARAMETERS AS NECESSARY. REGISTER ASSIGNMENT FOR THE
066F 1889 : PARAMETERS ARE AS FOLLOWS:
066F 1890 :
066F 1891 : R5 = ACCESS MODE.
066F 1892 : R9 = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
066F 1893 : R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
066F 1894 :
066F 1895 :
066F 1896 EXEDELLNM:
066F 1897
5A 08 AC D0 066F 1898 MOVL DNLOGNAM(AP),R10 ;ADDRESS OF LOGICAL NAME DESCRIPTOR
15 13 0673 1899 BEQL 10$ ;BRANCH IF NOT PRESENT
      0675 1900 IFNORD #8,(R10),900$ ;CHECK ACCESS TO DESCRIPTOR
      067B 1901 MOVAQ (R10),RO ;ADDRESS OF DESCRIPTOR
      067E 1902 BSBW LNMSPROBER ;PROBE LOGICAL NAME STRING
      EA 50 E9 0681 1903 BLBC R0,930$ ;BRANCH IF CAN'T READ OR NOT PRESENT
      7E 51 7D 0684 1904 MOVQ R1,-(SP) ;SAVE LOGICAL NAME DESCRIPTOR
      5A 5E D0 0687 1905 MOVL SP,R10 ;ADDRESS OF LOGICAL NAME DESCRIPTOR
      068A 1906 10$: MOVL DNTABNAM(AP),R9 ;ADDRESS OF TABLE NAME DESCRIPTOR
      DB 13 068E 1908 BEQL 910$ ;ERROR IF NOT PRESENT
      0690 1909 IFNORD #8,(R9),900$ ;CHECK ACCESS TO DESCRIPTOR
      0696 1910 MOVAQ (R9),RO ;SET UP CALL
      F964' 30 0699 1911 BSBW LNMSPROBER ;PROBE TABLE NAME STRING
      CF 50 E9 069C 1912 BLBC R0,930$ ;BRANCH IF CAN'T READ OR NOT PRESENT
      7E 51 7D 069F 1913 MOVQ R1,-(SP) ;SAVE TABLE NAME DESCRIPTOR
      59 5E D0 06A2 1914 MOVL SP,R9 ;ADDRESS OF TABLE NAME DESCRIPTOR
      06A5 1915
50 0C AC D0 06A5 1916 MOVL DNACMODE(AP),RO ;GET SPECIFIED ACCESS MODE
14 13 06A9 1917 BEQL 20$ ;BRANCH IF NOT PRESENT
      06AB 1918 IFNORD #1,(R0),900$ ;CHECK ACCESS TO VALUE
      06B1 1919 MOVZBL (R0),RO ;GET VALUE
      50 03 D1 06B4 1920 CMPL #PSL$C_USER,RO ;CHECK FOR VALID ACCESS MODE
      B2 1F 06B7 1921 BLSSU 910$ ;INVALID ACCESS MODE
      06B9 1922
      06B9 1923 :
      06B9 1924 : IF THE ACCESS MODE OF THE LOGICAL NAME TABLE ENTRY(S) TO BE DELETED WAS
      06B9 1925 : EXPLICITELY SPECIFIED AND THE CALLER HAS THE SYSNAM PRIVILEGE, THEN THE
      06B9 1926 : SPECIFIED ACCESS MODE IS USED TO QUALIFY THE LOGICAL NAME TABLE ENTRY(S)
      06B9 1927 : DELETION. OTHERWISE, THE ACCESS MODE OF THE CALLER IS MAXIMIZED WITH ANY
      06B9 1928 : EXPLICITELY SPECIFIED ACCESS MODE AND USED TO QUALIFY THE LOGICAL NAME TABLE
      06B9 1929 : ENTRY(S) DELETION.
      06B9 1930 :
      06B9 1931
      06B9 1932 IFPRIV SYSNAM,30$ ;CALLER HAS SYSNAM PRIVILEGE

```

00000000'GF 16 06BF 1933 20\$: JSB G^EXESMAXACMODE :MAXIMIZED WITH MODE OF CALLER TO
 55 50 DO 06C5 1934 30\$: MOVL R0,R5 ;DETERMINE ACCESS MODE OF TABLE ENTRY(S)
 06C8 1935
 06C8 1936 :
 06C8 1937 : RAISE IPL TO AST DELIVERY LEVEL SO THAT THERE ARE NO INTERRUPTIONS WHILE
 06C8 1938 : THE DELETION OF THE LOGICAL NAME TABLE ENTRY(S) IS BEING CARRIED OUT, AND
 06C8 1939 : LOCK THE LOGICAL NAME MUTEX FOR WRITING.
 06C8 1940 :
 06C8 1941 :
 F932' 30 06C3 1942 SETIPL S^#IPL\$ ASTDEL :RAISE TO AST DELIVERY LEVEL
 06CB 1943 BSBW LNMSLOCKRW ;LOCK TABLES FOR WRITING
 06CE 1944
 06CE 1945 :
 06CE 1946 : IF A LOGICAL NAME WAS SPECIFIED, THEN IT REPRESENTS THE LOGICAL NAME TABLE
 06CE 1947 : ENTRY WITH THE SPECIFIED ACCESS MODE WHICH IS TO BE DELETED FROM THE LOGICAL
 06CE 1948 : NAME TABLE WITH THE SPECIFIED LOGICAL NAME TABLE NAME. FIND THE FIRST SUCH
 06CE 1949 : LOGICAL NAME TABLE ENTRY AND DELETE IT. IF THE ACCESS MODE REQUESTED IS OTHER
 06CE 1950 : THAN USER, THAN ALL OUTER ACCESS MODE ALIASES IN THIS TABLE ARE ALSO DELETED.
 06CE 1951 :
 06CE 1952 : NOTE THAT IF THE FIRST SUCH LOGICAL NAME TABLE ENTRY IS THAT OF A LOGICAL
 06CE 1953 : NAME TABLE, THEN ALL LOGICAL NAMES CONTAINED WITHIN THE TABLE TOGETHER WITH
 06CE 1954 : ALL ITS SUBTABLES AND THE LOGICAL NAMES CONTAINED WITHIN THEM ARE DELETED.
 06CE 1955 :
 06CE 1956 :
 5A D5 06CE 1957 TSTL R10 :IS A SPECIFIC ENTRY TO BE DELETED?
 54 13 06D0 1958 BEQL 60\$;BRANCH IF SUPPOSED TO DELETE THEM ALL
 06D2 1959
 50 6A 7D 06D2 1960 MOVQ (R10),R0 :LOGICAL NAME DESCRIPTOR
 52 69 7D 06D5 1961 MOVQ (R9),R2 ;TABLE NAME DESCRIPTOR
 F925' 30 06D8 1962 BSBW LNMS\$SEARCHLOG ;SEARCH FOR JUST SUCH A LOGICAL NAME
 67 50 E9 06DB 1963 BLBC R0,80\$;EXIT IF NO NAME FOUND
 06DE 1964
 0B A1 55 91 06DE 1965 CMPB R5_LNMB\$B_ACMODE(R1) :DO THE ACCESS MODES MATCH
 07 13 06E2 1966 BEQLU 40\$;IF SO THEN HAVE FOUND ENTRY TO DELETE
 50 01BC 8F 3C 06E4 1967 MOVZWL #\$\$\$_NOLOGNAM,R0 ;OTHERWISE GO RETURN ERROR
 5A 11 06E9 1968 BRB 80\$
 06EB 1969
 06EB 1970 :
 06EB 1971 : DETERMINE WHETHER THE CALLER CAN DELETE THE LOGICAL NAME TABLE ENTRY, AND
 06EB 1972 : THEN DELETE IS IF SUCH ACCESS IS ALLOWED. THE ACCESS REQUIREMENTS FOR DELETION
 06EB 1973 : ARE AS FOLLOWS:
 06EB 1974 :
 06EB 1975 : LOGICAL NAME: THE CALLER NEEDS WRITE ACCESS TO THE CONTAINING LOGICAL
 06EB 1976 : NAME TABLE.
 06EB 1977 :
 06EB 1978 : LOGICAL NAME TABLE: THE CALLER NEEDS EITHER WRITE ACCESS TO THE CONTAINING
 06EB 1979 : LOGICAL NAME TABLE (THE SYSTEM OR PROCESS DIRECTORY
 06EB 1980 : TABLE), OR DELETE ACCESS TO THE LOGICAL NAME TABLE
 06EB 1981 : ITSELF.
 06EB 1982 :
 06EB 1983 : NOTE THAT IF THE LOGICAL NAME TABLE ENTRY IS PROCESS-PRIVATE THEN NO
 06EB 1984 : PROTECTION CHECKING IS PERFORMED, AND THE CALLER CAN ALWAYS DELETE THE ENTRY.
 06EB 1985 :
 06EB 1986 :
 32 51 1F E1 06EB 1987 40\$: BBC #31,R1,50\$:SKIP CHECK IF PROCESS-PRIVATE ENTRY
 52 02 9A 06EF 1988 MOVZBL #WRITE_ACCESS,R2 ;CODE FOR ACCESS CHECK
 51 DD 06F2 1989 PUSHL R1 ;SAVE LNMB ADDRESS

```

51 0C A1  D0 06F4 1990      MOVL   LNMB$L_TABLE(R1),R1    ;GET TABLE HEADER ADDRESS
F905' 30 06F8 1991      BSBW   LNMSCHECK_PROT   ;CHECK THE PROTECTION
51 8ED0 06FB 1992      POPL   R1                 ;RESTORE LNMB
20 50 E8 06FE 1993      BLBS   R0,50$            ;DELETE THE ENTRY ON SUCCESS
03 E1 0701 1994      BBC    #LNMB$V_TABLE,-  ;RETURN AN ERROR ON FAILURE IF THE
3F 10 A1 0703 1995      LNMB$B_FLAGS(R1),80$ ;ENTRY TO BE DELETED IS NOT A TABLE
                  0706 1996
52 08 9A 0706 1997      MOVZBL #DELETE_ACCESS,R2 ;CODE FOR ACCESS CHECK
51 DD 0709 1998      PUSHL  R1                 ;SAVE LNMB ADDRESS
50 11 A1 9A 070B 1999      MOVZBL LNMB$T_NAME(R1),R0 ;RETRIEVE SIZE OF NAME STRING
51 12 A140 9E 070F 2000      MOVAB  LNMB$T_NAME+1(R1)[R0],R1 ;POSITION TO LNMX
51 05 A1 9E 0714 2001      MOVAB  LNMX$T_XLATION+1(R1),R1 ;POSITION TO TABLE HEADER
F8E5' 30 0718 2002      BSBW   LNMSCHECK_PROT ;CHECK THE PROTECTION
51 8ED0 071B 2003      POPL   R1                 ;RESTORE LNMB
24 50 E9 071E 2004      BLBC   R0,80$            ;QUIT ON FAILURE
                  0721 2005
F8DC' 30 0721 2006      50$:    BSBW   LNMSDELETE_LNMB ;DELETE ALL APPROPRIATE LOGICAL NAMES
1F 11 0724 2007      BRB    80$                ;AND RETURN STATUS OF DELETION
                  0726 2008
                  0726 2009 :
                  0726 2010 : IF NO LOGICAL NAME WAS SPECIFIED, THEN ALL THE LOGICAL NAME TABLE ENTRIES
                  0726 2011 : CONTAINED WITHIN THE SPECIFIED LOGICAL NAME TABLE AND POSSESSING AN ACCESS
                  0726 2012 : MODE EQUAL TO OR GREATER THAN THE DESIGNATED ACCESS MODE ARE TO BE DELETED.
                  0726 2013 : POSITION TO THE SPECIFIED LOGICAL NAME TABLE, AND PERFORM THE ACTUAL DELETION
                  0726 2014 : OF THE NAMES CONTAINED WITHIN THE TABLE.
                  0726 2015 :
                  0726 2016 : TWO POSSIBLE CASES EXIST:
                  0726 2017 :
                  0726 2018 : CASE 1: THE SPECIFIED LOGICAL NAME TABLE NAME IS THAT OF A LOGICAL NAME TABLE
                  0726 2019 : OTHER THAN THE SYSTEM/PROCESS DIRECTORY TABLE. IN THIS CASE ALL THE
                  0726 2020 : LOGICAL NAMES, BUT NOT THE LOGICAL NAME TABLE SUB-STRUCTURE, THAT
                  0726 2021 : POSSESS AN ACCESS MODE EQUAL TO OR GREATER THAN THE DESIGNATED ACCESS
                  0726 2022 : MODE ARE DELETED. THE LOGICAL NAME TABLE SUB-STRUCTURE IS NOT DELETED
                  0726 2023 : BECAUSE ALL LOGICAL NAME TABLES ARE CONTAINED WITHIN THE SYSTEM OR
                  0726 2024 : PROCESS DIRECTORY TABLES AND NOT WITHIN INDIVIDUAL LOGICAL NAME
                  0726 2025 : TABLES.
                  0726 2026 :
                  0726 2027 : CASE 2: THE SPECIFIED LOGICAL NAME TABLE NAME IS THAT OF THE SYSTEM/PROCESS
                  0726 2028 : DIRECTORY TABLE. IN THIS CASE ALL OF THE LOGICAL NAME TABLE ENTRIES
                  0726 2029 : (INCLUDING ALL LOGICAL NAME TABLES) CONTAINED WITHIN THE SPECIFIED
                  0726 2030 : DIRECTORY TABLE AND POSSESSING AN ACCESS MODE EQUAL TO OR GREATER
                  0726 2031 : THAN THE DESIGNATED ACCESS MODE ARE DELETED.
                  0726 2032 :
                  0726 2033 : NOTE THAT IF THE LOGICAL NAME TABLE IS PROCESS-PRIVATE THEN NO PROTECTION
                  0726 2034 : CHECKING IS PERFORMED, AND THE CALLER CAN ALWAYS DELETE THE ENTRY(S) IN THE
                  0726 2035 : TABLE.
                  0726 2036 :
                  0726 2037 :
51 55  D0 0726 2038 60$:    MOVL   R5,R1             ;ACCESS MODE
52 69  7D 0729 2039      MOVQ   (R9),R2           ;TABLE NAME DESCRIPTOR
F8D1' 30 072C 2040      BSBW   LNMSFIRSTTAB   ;SEARCH FOR SPECIFIED LOGICAL NAME TABLE
13 50  E9 072F 2041      BLBC   R0,80$            ;BRANCH ON FAILURE
                  0732 2042
                  0732 2043 :
                  0732 2044 : MAKE SURE THAT THE CALLER HAS WRITE ACCESS TO THE LOGICAL NAME TABLE BEFORE
                  0732 2045 : DELETING ALL THE ENTRIES THAT IT CONTAINS.
                  0732 2046 :

```

00 00 E1 0732 2047
09 61 0734 2048 BBC #LNMTHSV SHAREABLE,-
52 02 9A 0736 2050 MOVZBL #WRITE ACCESS,R2
F8C4' 30 0739 2051 BSBW LNMSCHECK_PROT
06 50 E9 073C 2052 BLBC R0,80\$
52 55 D0 073F 2053 :
F8BB' 30 0742 2054 70\$: MOVL R5,R2
0745 2055 BSBW LNMSDELETE_TAB ;ACCESS MODE
0745 2056 :
0745 2057 :
0745 2058 : UNLOCK THE LOGICAL NAME TABLE MUTEX AND RETURN STATUS TO CALLER.
0745 2059 :
0745 2060 :
50 DD 0745 2061 80\$: PUSHL R0
F8B6' 30 0747 2062 BSBW LNMSUNLOCK ;SAVE RETURN CODE OVER UNLOCKING
50 8ED0 074A 2063 POPL R0 ;UNLOCK TABLES
04 074D 2064 RET ;RESTORE RETURN CODE
074E 2065 :
074E 2066 .DISABLE LSB
074E 2067 :
074E 2068 ; .PAGE
;

074E 2070 .SBTTL EXE\$TRNLNM - TRANSLATE LOGICAL NAME
074E 2071 +
074E 2072 EXE\$TRNLNM - TRANSLATE LOGICAL NAME
074E 2073
074E 2074 THIS SERVICE PROVIDES THE CAPABILITY TO LOOKUP A LOGICAL NAME IN THE SPECIFIED
074E 2075 LOGICAL NAME TABLE, AND TO RETURN INFORMATION ABOUT IT IN THE CALLER SPECIFIED
074E 2076 ITEM LIST.
074E 2077
074E 2078 INPUTS:
074E 2079
074E 2080 TRATTR(AP) = ADDRESS OF LOGICAL NAME TRANSLATION ATTRIBUTES.
074E 2081 TRTABNAM(AP) = ADDRESS OF TABLE NAME STRING DESCRIPTOR.
074E 2082 TRLOGNAM(AP) = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR.
074E 2083 TRACMODE(AP) = ADDRESS OF ACCESS MODE.
074E 2084 TRITMLST(AP) = ADDRESS OF ITEM LIST.
074E 2085
074E 2086 R4 = CURRENT PROCESS PCB ADDRESS.
074E 2087
074E 2088 OUTPUTS:
074E 2089
074E 2090 R0 LOW BIT CLEAR INDICATES FAILURE TO TRANSLATE LOGICAL NAME.
074E 2091
074E 2092 R0 = SSS_ACCVIO - LOGICAL NAME DESCRIPTOR, LOGICAL NAME STRING,
074E 2093 TABLE NAME DESCRIPTOR, TABLE NAME STRING, AN ITEM IN THE
074E 2094 ITEM LIST, AN INDEX ITEM BUFFER CANNOT BE READ BY
074E 2095 CALLING ACCESS MODE. A TABLE ITEM BUFFER, A TABLE ITEM
074E 2096 SIZE BUFFER, AN ATTRIBUTES ITEM BUFFER, AN ATTRIBUTES
074E 2097 ITEM SIZE BUFFER, A STRING ITEM BUFFER, A STRING ITEM
074E 2098 SIZE BUFFER, A LENGTH ITEM BUFFER, A LENGTH ITEM SIZE
074E 2099 BUFFER, AN ACMODE ITEM BUFFER, AN ACMODE ITEM SIZE
074E 2100 BUFFER CANNOT BE WRITTEN BY CALLING ACCESS MODE.
074E 2101
074E 2102 R0 = SSS_BADPARAM - INVALID ATTRIBUTE, ACCESS MODE, ITEM TYPE,
074E 2103 ITEM LENGTH, SPECIFIED. LOGICAL NAME DESCRIPTOR OR
074E 2104 TABLE NAME DESCRIPTOR NOT SPECIFIED.
074E 2105
074E 2106 R0 = SSS_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
074E 2107 LOGICAL OR TABLE NAME STRING SPECIFIED.
074E 2108
074E 2109 R0 = SSS_IVLOGTAB - INVALID TABLE NAME SPECIFIED.
074E 2110
074E 2111 R0 = SSS_NOLOGNAM - LOGICAL TABLE NAME ENTRY SPECIFIED DOES NOT
074E 2112 EXIST.
074E 2113
074E 2114 R0 = SSS_NOPRIV - PROCESS DOES NOT HAVE PRIVILEGE TO ACCESS
074E 2115 THE SPECIFIED LOGICAL NAME TABLE ENTRY.
074E 2116
074E 2117 R0 = SSS_TOOMANYLNM - TOO MANY LEVELS OF RECURSION IN SEARCH
074E 2118 FOR LOGICAL NAME TABLE.
074E 2119
074E 2120 R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
074E 2121
074E 2122 R0 = SSS_BUFFEROVF - REQUEST SUCCESSFULLY COMPLETED. AN ITEM
074E 2123 BUFFER IS NOT LARGE ENOUGH TO HOLD REQUESTED DATA.
074E 2124
074E 2125 R0 = SSS_NORMAL - NORMAL COMPLETION.
074E 2126

074E 2127 : SIDE EFFECTS:
 074E 2128 :
 074E 2129 :
 074E 2130 :
 074E 2131 :-
 074E 2132 :
 0742 0FFC 000F 0000000F 2133 .PSECT YSEXEPAGED
 31 0011 2134 .ENTRY EXESTRLNM,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
 0014 2135 BRW EXESTRLNM
 0000074E 2136 .PSECT YFSSLNM
 50 0C 3C 074E 2137 .ENABLE LSB
 04 0751 2140 MOVZWL #SSS_ACCVIO,RO ;ACCESS VIOLATION
 50 14 3C 0752 2141 910\$: MOVZWL #SSS_BADPARAM,RO ;BAD SYSTEM SERVICE PARAMETER
 04 0755 2142 920\$: RET
 0756 2143 EXESTRLNM:
 0756 2144
 0756 2145
 0756 2146 : VALIDATE AND COPY PARAMETERS AS NECESSARY. REGISTER ASSIGNMENT FOR THE
 0756 2147 : PARAMETERS ARE AS FOLLOWS:
 0756 2148 :
 0756 2149 : R5 = ACCESS MODE.
 0756 2150 : R7 = ATTRIBUTE BITS.
 0756 2151 : R9 = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
 0756 2152 : R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
 0756 2153 : R11 = ADDRESS OF START OF THE ITEM LIST.
 0756 2154 :
 0756 2155 :
 0756 2156 :---
 0756 2157 :
 7E D4 0756 2158 CLRL -(SP) ;THIS WILL EVENTUALLY BE CHAIN COUNTER
 0758 2159
 57 04 AC 0D 0758 2160 MOVL TRATTR(AP),R7 ;TRANSLATION ATTRIBUTES
 12 13 075C 2161 BEQL 10\$;BRANCH IF NOT PRESENT
 57 FDFFFFFF 57 67 0D 0764 2162 IFNORD #4,(R7).900\$;CHECK ACCESS
 8F D3 0767 2163 MOVL (R7),R7 ;GET VALUE
 076E 2164 BITL #^C<-
 076E 2165 LNMSM_CASE_BLIND - ;CASE SENSITIVE VS. CASE INSENSITIVE
 076E 2166 >R7
 E2 12 076E 2167 BNEQ 910\$;INVALID TRANSLATION ATTRIBUTES
 0770 2168
 5A 0C AC 0D 0770 2169 10\$: MOVL TRLOGNAM(AP),R10 ;ADDRESS OF LOGICAL NAME DESCRIPTOR
 DC 13 0774 2170 BEQL 910\$;ERROR IF NOT PRESENT
 0776 2171 IFNORD #8,(R10).900\$;CHECK ACCESS TO DESCRIPTOR
 50 6A. 7E 077C 2172 MOVAQ (R10),R0 ;ADDRESS OF DESCRIPTOR
 F87E. 30 077F 2173 BSBW LNMSPROBER ;PROBE LOGICAL NAME STRING
 DO 50 E9 0782 2174 BLBC R0,920\$;BRANCH IF CAN'T READ OR NOT PRESENT
 7E 51 7D 0785 2175 MOVQ R1,-(SP) ;SAVE LOGICAL NAME DESCRIPTOR
 5A 5E 0D 0788 2176 MOVL SP,R10 ;ADDRESS OF LOGICAL NAME DESCRIPTOR
 0788 2177
 59 08 AC 0D 0788 2178 MOVL TRTABNAM(AP),R9 ;ADDRESS OF TABLE NAME DESCRIPTOR
 C1 13 078F 2179 BEQL 910\$;ERROR IF NOT PRESENT
 0791 2180 IFNORD #8,(R9).900\$;CHECK ACCESS TO DESCRIPTOR
 50 69. 7E 0797 2181 MOVAQ (R9),R0 ;ADDRESS OF DESCRIPTOR
 F863. 30 079A 2182 BSBW LNMSPROBER ;PROBE TABLE NAME STRING
 B5 50 E9 079D 2183 BLBC R0,920\$;BRANCH IF CAN'T READ OR NOT PRESENT

7E 51 7D 07A0 2184
 59 5E D0 07A3 2185
 59 5E 07A6 2186
 50 55 03 9A 07A6 2187
 10 AC D0 07A9 2188
 0E 13 07AD 2189
 55 60 9A 07B5 2190
 55 03 D1 07B8 2191
 95 1F 07BB 2192
 07BD 2193
 07BD 2194
 07BD 2195
 07BD 2196 : RAISE IPL TO AST DELIVERY LEVEL SO THAT THERE ARE NO INTERRUPTIONS WHILE THE
 07BD 2197 : TRANSLATION OF THE LOGICAL NAME IS BEING CARRIED OUT. LOCK THE LOGICAL NAME
 07BD 2198 : MUTEX FOR READING, AND THEN SEARCH FOR THE SPECIFIED LOGICAL NAME.
 07BD 2199 :
 07BD 2200
 F83D' 30 07C0 2201 2000\$: SETIPL S^#IPL\$ ASTDEL
 54 DD 07C3 2202 BSBW LNMSLOCKR
 07C5 2203 PUSHL R4
 07C5 2204
 50 6A 7D 07C5 2205 MOVQ (R10), R0
 52 69 7D 07C8 2206 MOVQ (R9), R2
 04 57 19 E1 07CB 2207 BBC #LNMS\$V CASE_BLIND,R7,2010\$:CASE INSENSITIVE SEARCH?
 00 55 08 E2 07CF 2208 BBSS #8,R5,2010\$;YES - SET CORRESPONDING BIT
 07D3 2209
 07D3 2210 2010\$:
 00000002 07D3 2211 IF NE CAS_MEASURE
 00000000'EF D6 07D3 2212 INCL L^PMSSGL_LOGNAM
 07D9 2213 .ENDC
 07D9 2214
 F824' 30 07D9 2215 BSBW LNMSSEARCHLOG
 50 DD 07DC 2216 PUSHL R0
 23 50 E9 07DE 2217 BLBC R0,3020\$
 07E1 2218
 07E1 2219 :
 07E1 2220 : CHECK THE CALLER'S ACCESS TO THE LOGICAL TABLE THAT HAS BEEN FOUND TO CONTAIN
 07E1 2221 : THE SPECIFIED LOGICAL NAME. NOTE THAT ACCESS CHECKING IS NOT REQUIRED IF THE
 07E1 2222 : LOGICAL NAME IS A PROCESS-PRIVATE LOGICAL NAME.
 07E1 2223 :
 07E1 2224
 16 58 51 D0 07E1 2225 MOVL R1,R8
 51 51 1F E1 07E4 2226 BBC #31,R1,3010\$
 52 01 DC 07E8 2227 MOVL #READ_ACCESS,R2
 54 04 AE DO 07EB 2228 MOVL 4(SP),R4
 51 0C A8 DO 07EF 2229 MOVL LNMBSL_TABLE(R8),R1
 F80A' 30 07F3 2230 BSBW LNMSCHECK_PROT
 05 50 E8 07F6 2231 BLBS R0,3010\$
 6E 50 DO 07F9 2232 MOVL R0,(SP)
 06 11 07FC 2233 BRB 3020\$
 07FE 2234
 07FE 2235 :
 07FE 2236 : PROCESS THE ITEM LIST RETURNING WHAT INFORMATION THE CALLER WANTS INTO THE
 07FE 2237 : SPECIFIED ITEM BUFFERS.
 07FE 2238 :
 07FE 2239 : VALID ITEMS ARE:
 07FE 2240 : ACMODE - ACCESS MODE OF LOGICAL NAME TABLE ENTRY.

07FE	2241	:	ATTRIBUTES	- ATTRIBUTES OF LOGICAL NAME AND CURRENT TRANSLATION.
07FE	2242	:	INDEX	- INDEX OF TRANSLATION (INPUT ITEM).
07FE	2243	:	LENGTH	- LENGTH OF CURRENT TRANSLATION STRING.
07FE	2244	:	MAX INDEX	- MAXIMUM INDEX IN LOGICAL NAME TABLE ENTRY.
07FE	2245	:	STRING	- TRANSLATION STRING.
07FE	2246	:	TABLE	- LOGICAL NAME TABLE NAME STRING.
07FE	2247	:	PARENT	- PARENT LOGICAL NAME TABLE NAME STRING.
07FE	2248	:		
07FE	2249	:		
07FE	2250	:	REGISTER USAGE IS AS FOLLOWS:	
07FE	2251	:		
07FE	2252	:	R6 = CURRENT TRANSLATION INDEX.	
07FE	2253	:	R7 = ADDRESS OF FIRST LOGICAL NAME TRANSLATION BLOCK	
07FE	2254	:	R8 = ADDRESS OF LOGICAL NAME TABLE ENTRY.	
07FE	2255	:	R9 = ADDRESS OF CURRENT LOGICAL NAME TRANSLATION BLOCK.	
07FE	2256	:	R11 = ADDRESS OF CURRENT ITEM IN ITEM LIST.	
07FE	2257	:		
07FE	2258	:		

5B	14 AC	D0	07FE	2259	3010\$: MOVL TRITMLST(AP),R11	:ADDRESS OF ITEM LIST
	03	12	0802	2260	BNEQ 3030\$:PROCESS ITEM LIST IF THERE IS ONE
	01F1	31	0804	2261	3020\$: BRW 4010\$:BRANCH IF NONE
59	50 11 AB	9A	0807	2263	3030\$: MOVZBL LNMBST_NAME(R8),R0	:SIZE OF LOGICAL NAME STRING
	12 A840	9E	080B	2264	MOVAB LNMBST_NAME+1(R8)[R0],R9	:ADDRESS OF FIRST TRANSLATION BLOCK
	57 59	D0	0810	2265	MOVL R9,R7	:SAVE THIS ADDRESS
	56 D4	0813	2266		CLRL R6	:DEFAULT INDEX IS 0
	02 E0	0815	2267	3035\$: BBS #LNMX\$V XEND,-	:LAST TRANSLATION BLOCK?	
	10 69	0817	2268		LNMX\$B_FLAGS(R9),3038\$:YES - THEN POSITION TO GET ITEM
	01 A9	95	0819	2269	TSTB LNMX\$B_INDEX(R9)	:POSITION TO (OR PAST) 0 LNMX?
	0B 18	081C	2270		BGEQ 3038\$:YES - THEN POSITION TO GET ITEM
59	50 04 A9	9A	081E	2271	MOVZBL LNMX\$T_XLATION(R9),R0	:SIZE OF TRANSLATION STRING
	05 A940	9E	0822	2272	MOVAB LNMX\$T_XLATION+1(R9)[R0]	:POSITION TO NEXT TRANSLATION BLOCK
	EC 11	0827	2273		BRB 3035\$:CONTINUE LOOKING FOR SPECIFIED LNMX
		0829	2274	3038\$: IFNORD #4,(R11),3060\$:CHECK IF FIRST LONGWORD READABLE	
		082F	2275	3040\$: MOVZWL 2(R11),R2	:GET ITEM TYPE	
52	02 AB	3C	082F	2276	BEQL 3020\$:DONE IF ITEM TYPE IS ZERO
	CF	13	0833	2277	IFNORD #12,4(R11),3060\$:CHECK REST OF THIS DESCRIPTOR
		0835	2278			:PLUS FIRST LONGWORD OF NEXT ONE
		083C	2279		CASE R2,<-	:HANDLE EACH ITEM TYPE SEPARATELY
		083C	2280		3100\$, -	:INDEX ITEM
		083C	2281		3200\$, -	:STRING ITEM
		083C	2282		3300\$, -	:ATTRIBUTES ITEM
		083C	2283		3400\$, -	:TABLE ITEM
		083C	2284		3500\$, -	:LENGTH ITEM
		083C	2285		3600\$, -	:ACMODE ITEM
		083C	2286		3700\$, -	:MAX INDEX ITEM
		083C	2287		3800\$, -	:PARENT ITEM
		083C	2288		> W #1	
		083C	2289		R2, #^XLNMS_CHAIN	:CHECK FOR CHAINED ITEM LIST
FFFF	8F 52	B1	0850	2291	CMPW BEQL 3070\$	
	06 13	0855	2292		BRW 3980\$:ILLEGAL ITEM
019B	31	0857	2293	3050\$: BRW 3970\$:ACCVIO	
0193	31	085A	2294	3060\$: BRW 3970\$		
		085D	2295			
		085D	2296			
		085D	2297		: PROCESS A CHAINED ITEM LIST.	

			085D	2298 :			
0400	8F	FC AD	D6	085D	2299		
		FC AD	B1	0860	2300	3070\$: INCL -4(FP)	
		EF	14	0866	2301	CMPW -4(FP),#1024	
5B	04	AB	CO	0868	2302	BGTR 3050\$	
		BB	11	086C	2303	MOVL 4(R11),R11	
				086E	2304	BRB 3038\$	
				086E	2305		
				086E	2306		
				086E	2307	: INDEX ITEM.	
				086E	2308		
				086E	2309	: VALIDATE THE ITEM. POSITION TO THE TRANSLATION BLOCK WITH THE SAME	
				086E	2310	: OR A GREATER INDEX THEN THAT SPECIFIED BY THE ITEM.	
				086E	2311		
				086E	2312		
6B	04	B1	086E	2313	3100\$: CMPW #4(R11)	: IS ITEM BUFFER AT LEAST A LONGWORD?	
		E4	1A	0871	2314	BGTRU 3050\$: ERROR IF NOT
56	04	AB	DO	0873	2315	MOVL 4(R11),R6	: PICK UP THE ITEM BUFFER ADDRESS
				0877	2316	IFNORD (R11),(R6),3060\$: PROBE IT FOR READABILITY
56	66	F6	087D	2317	CVTLB (R6),R6	: IS THE INDEX BETWEEN -127 AND 127?	
		D5	1D	0880	2318	BVS 3050\$: NO - INVALID INDEX ITEM
56	56	9A	0882	2319	MOVZBL R6,R6	: ZERO OUT HIGH ORDER THREE BYTES	
59	57	DO	0885	2320	MOVL R7,R9	: ADDRESS OF FIRST TRANSLATION BLOCK	
			0888	2321			
		02	E0	0888	2322	3110\$: BBS #LNMX\$V XEND,-	: LAST TRANSLATION BLOCK?
01	A9	11	69	088A	2323	LNMXSB FLAGS(R9),3120\$: YES - THEN POSITION TO NEXT ITEM
		56	91	088C	2324	CMPB R6,LNMXSB_INDEX(R9)	: POSITION TO (OR PAST) REQUESTED LNMX?
		0B	15	0890	2325	BLEQ 3120\$: YES - THEN POSITION TO NEXT ITEM
50	04	A9	9A	0892	2326	MOVZBL LNMXST_XLATION(R9),R0	: SIZE OF TRANSLATION STRING
59	05	A940	9E	0896	2327	MOVAB LNMXST_XLATION+1(R9)[R0],R9	: POSITION TO NEXT TRANSLATION BLOCK
		EB	11	0898	2328	BRB 3110\$: CONTINUE LOOKING FOR SPECIFIED LNMX
		014A	31	089D	2329	3120\$: BRW 3960\$: GO POSITION TO NEXT ITEM
				08A0	2330		
				08A0	2331		
				08A0	2332	: STRING ITEM.	
				08A0	2333		
				08A0	2334	: VALIDATE THE ITEM. IF THE CURRENT TRANSLATION BLOCK IS THE LAST	
				08A0	2335	: TRANSLATION BLOCK, OR IF THE INDEX OF THE CURRENT TRANSLATION BLOCK	
				08A0	2336	: DOES NOT MATCH THE INDEX LAST SPECIFIED, THEN NOTHING IS RETURNED	
				08A0	2337	: (A ZERO IS STORED IN THE RETURN ADDRESS LENGTH BUFFER). OTHERWISE,	
				08A0	2338	: THE EQUIVALENCE STRING OF THE SPECIFIED TRANSLATION IS RETURNED.	
				08A0	2340		
		50	7C	08A0	2341	3200\$: CLRQ R0	: ASSUME 0 BYTES WILL BE RETURNED
		02	E0	08A2	2342	BBS #LNMX\$V XEND,-	: POSITION TO LAST LNMX?
01	A9	OD	69	08A4	2343	LNMXSB FLAGS(R9),3210\$: YES - 0 BYTES WILL BE RETURNED
		56	91	08A6	2344	CMPB R6,LNMXSB_INDEX(R9)	: POSITIONED TO SPECIFIED LNMX?
		07	12	08AA	2345	BNEQ 3210\$: NO - 0 BYTES WILL BE RETURNED
51	04	A9	9E	08AC	2346		
50	81	90	80B0	2347	MOVAB LNMXST_XLATION(R9),R1	: RETRIEVE ADDRESS AND SIZE OF	
00BF		31	08B3	2348	MOVBL (R1)+,R0	: TRANSLATION STRING	
			08B6	2349	3210\$: BRW 3900\$: GO MOVE STRING INTO CURRENT ITEM	
			08B6	2350			
			08B6	2351			
			08B6	2352	: ATTRIBUTES ITEM.		
			08B6	2353			
			08B6	2354	: VALIDATE THE ITEM. THE ATTRIBUTES OF THE LOGICAL NAME BLOCK ARE		

08B6 2355 : ALWAYS RETURNED. IF A TRANSLATION WITH THE CURRENT INDEX EXISTS
 08B6 2356 : THEN THE TRANSLATION ATTRIBUTES ARE RETURNED AND LNMSV_EXIST IS SET.
 08B6 2357 :
 08B6 2358 :

52 10 A8 9A 08B6 2359 3300\$: MOVZBL LNMBSB_FLAGS(R8),R2 ;LOGICAL NAME BLOCK FLAGS
 02 E0 08BA 2360 BBS #LNMX\$V_XEND,- POSITION TO LAST LNMX?
 17 69 08BC 2361 LNMXSB_FLAGS(R9),3310S YES - TRANSLATION DOES NOT EXIST
 01 A9 56 91 08BE 2362 CMPB R6_LNMXSB_INDEX(R9) POSITIONED TO SPECIFIED LNMX?
 11 12 08C2 2363 BNEQ 3310S NO - TRANSLATION DOES NOT EXIST

50 50 69 9A 08C4 2364 MOVZBL LNMXSB_FLAGS(R9),R0 ;TRANSLATION BLOCK FLAGS
 50 50 08 9C 08C7 2365 ROTL #8,R0,R0 ROTATE THEM INTO THEIR PROPER PLACE
 52 50 C8 08CB 2366 BISL2 R0,R2 STORE THEM
 52 00000400 8F C8 08CE 2368 BISL2 #LNMSM_EXISTS,R2 SET THE TRANSLATION DOES EXIST BIT

51 0C A8 D0 08D5 2370 3310\$: MOVL LNMBSL_TABLE(R8),R1 ;GET CONTAINING TABLE HEADER ADDRESS
 00 E1 08D9 2371 BBC #LNMTHSV_SHAREABLE,- IS THE CONTAINING TABLE SHAREABLE?
 07 61 08DB 2372 LNMTHSB_FLAGS(R1),3320S GO MOVE ATTRIBUTES IF NOT
 52 00010000 8F C8 08DD 2373 BISL2 #LNMSM_SHAREABLE,R2 SET SHAREABLE BIT
 00E2 31 08E4 2374 BRW 3940S ;GO MOVE ATTRIBUTES INTO CURRENT ITEM

08E7 2375 :
 08E7 2376 : TABLE ITEM.
 08E7 2377 :
 08E7 2378 :
 08E7 2379 : VALIDATE THE ITEM. RETURN THE TABLE NAME STRING OF THE LOGICAL
 08E7 2380 : NAME TABLE IN WHICH THIS LOGICAL NAME TABLE ENTRY IS FOUND.
 08E7 2381 :
 08E7 2382 :
 51 0C A8 D0 08E7 2383 3400\$: MOVL LNMBSL_TABLE(R8),R1 ;ADDRESS OF TABLE HEADER
 51 09 A1 D0 08EB 2384 MOVL LNMTHSNAME(R1),R1 ;ADDRESS OF TABLE LNMB
 51 11 A1 9E 08EF 2385 MOVAB LNMBST_NAME(R1),R1 ;RETRIEVE ADDRESS AND SIZE OF
 50 81 9A 08F3 2386 MOVZBL (R1)+,R0 TABLE NAME
 007C 31 08F6 2387 BRW 3900S ;GO MOVE TABLE INTO CURRENT ITEM

08F9 2388 :
 08F9 2389 : LENGTH ITEM.
 08F9 2390 :
 08F9 2391 :
 08F9 2392 : VALIDATE THE ITEM. IF THE CURRENT TRANSLATION BLOCK IS THE LAST
 08F9 2393 : TRANSLATION BLOCK, OR IF THE INDEX OF THE CURRENT TRANSLATION
 08F9 2394 : BLOCK DOES NOT MATCH THE INDEX LAST SPECIFIED, THEN NOTHING IS
 08F9 2395 : RETURNED (A ZERO IS STORED IN THE RETURN ADDRESS LENGTH BUFFER).
 08F9 2396 : OTHERWISE, THE LENGTH OF THE TRANSLATION STRING OF THE SPECIFIED
 08F9 2397 : TRANSLATION IS RETURNED.
 08F9 2398 :
 08F9 2399 :
 52 D4 08F9 2400 3500\$: CLRL R2 ;ASSUME 0 BYTES WILL BE RETURNED

02 E0 08FB 2401 BBS #LNMX\$V_XEND,- POSITION TO LAST LNMX?
 0A 69 08FD 2402 LNMXSB_FLAGS(R9),3510S YES - GO RETURN 0 LENGTH
 01 A9 56 91 08FF 2403 CMPB R6_LNMXSB_INDEX(R9) POSITIONED TO SPECIFIED LNMX?
 04 12 0903 2404 BNEQ 3510S NO - GO RETURN 0 LENGTH
 52 04 A9 9A 0905 2406 MOVZBL LNMXST_XLATION(R9),R2 LENGTH OF TRANSLATION STRING
 00BD 31 0909 2407 3510\$: BRW 3940S MOVE LONGOWRD ITEM VALUE
 00E6 31 090C 2408 3520\$: BRW 3980S ;BADPARAM

090F 2409 :
 090F 2410 : ACMODE ITEM.
 090F 2411 :

090F 2412 :
 090F 2413 : VALIDATE THE ITEM. RETURN THE ACCESS MODE OF THE LOGICAL NAME TABLE
 090F 2414 : ENTRY.
 090F 2415 :
 090F 2416 :
 6B 01 B1 090F 2417 3600\$: CMPW #1,(R11)
 F8 1A 0912 2418 BGTRU 3520\$;IS ITEM BUFFER AT LEAST A BYTE?
 0914 2419 ;ERROR IF NOT
 50 04 AB D0 0914 2420 MOVL 4(R11),R0 ;RETRIEVE ADDRESS OF ITEM BUFFER
 0918 2421 IFNOWRT #1,(R0),3620\$;PROBE ITEM BUFFER
 60 0B A8 90 091E 2422 MOVB LNMBSB_ACMODE(R8),(R0) ;RETURN ACMODE IN CURRENT ITEM
 50 08 AB D0 0922 2423 MOVL 8(R11),R0 ;WAS A RETURN LENGTH BUFFER SPECIFIED?
 09 13 0926 2424 BEQL 3610\$;DON'T RETURN LENGTH IF ONE WASN'T
 60 01 B0 092E 2425 IFNOWRT #2,(R0),3620\$;PROBE RETURN LENGTH BUFFER
 00B6 31 0931 2426 MOVW #1,(R0) ;MOVE IN NUMBER OF BYTES RETURNED
 00B9 31 0934 2427 3610\$: BRW 3960\$;GO POSITION TO NEXT ITEM
 0937 2428 3620\$: BRW 3970\$;ACCVIO
 0937 2429 :
 0937 2430 :
 0937 2431 : MAX_INDEX ITEM.
 0937 2432 :
 0937 2433 : VALIDATE THE ITEM. RETURN THE MAXIMUM POSITIVE INDEX DEFINED FOR
 0937 2434 : THE LOGICAL NAME TABLE ENTRY.
 0937 2435 :
 0937 2436 :
 52 01 CE 0937 2437 3700\$: MNEGL #1,R2 ;INITIALIZE MAX INDEX TO -1
 51 57 D0 093A 2438 MOVL R7,R1 ;ADDRESS OF FIRST TRANSLATION BLOCK
 093D 2439 :
 52 02 E0 093D 2440 3710\$: BBS #LNMX\$V_XEND,- ;POSITION TO LAST LNMX?
 OF 61 093F 2441 LNMX\$B_FLAGS(R1),3720\$;YES - HAVE FOUND MAXIMUM INDEX
 52 01 A1 98 0941 2442 CVTBL LNMX\$B_INDEX(R1),R2 ;NO - SAVE INDEX
 50 04 A1 9A 0945 2443 MOVZBL LNMX\$T_XLATION(R1),R0 ;SIZE OF TRANSLATION STRING
 51 05 A140 9E 0949 2444 MOVAB LNMX\$T_XLATION+1(R1)[R0];R1 :POSITION TO NEXT TRANSLATION BLOCK
 ED 11 094E 2445 BRB 3710\$;CONTINUE LOOKING FOR LAST LNMX
 0950 2446 :
 52 D5 0950 2447 3720\$: TSTL R2 ;IS MAXIMUM INDEX A NEGATIVE NUMBER?
 75 18 0952 2448 BGEQ 3940\$;NO - GO RETURN MAXIMUM INDEX
 52 01 CE 0954 2449 MNEGL #1,R2 ;YES - SET MAXIMUM INDEX TO -1
 70 11 0957 2450 BRB 3940\$;GO MOVE MAXIMUM INDEX INTO CURRENT ITEM
 0959 2451 :
 0959 2452 :
 0959 2453 : PARENT TABLE ITEM.
 0959 2454 :
 0959 2455 : VALIDATE THE ITEM. IF THIS LNMB IS FOR A LOGICAL NAME TABLE, THEN RETURN
 0959 2456 : THE TABLE NAME STRING OF THIS TABLE'S PARENT LOGICAL NAME TABLE; OTHERWISE,
 0959 2457 : RETURN 0.
 0959 2458 :
 0959 2459 :
 50 7C 0959 2450 3800\$: CLRQ R0 ;ASSUME 0 BYTES WILL BE RETURNED
 03 E1 0958 2461 BBC #LNMB\$V_TABLE,- ;RETURN 0 BYTES IF THIS IS NOT A LNMB
 15 10 A8 095D 2462 LNMB\$B_FLAGS(R8),3900\$;FOR A LOGICAL NAME TABLE
 52 05 A7 9E 0960 2463 MOVAB LNMX\$T_XLATION+1(R7),R2 ;POSITION TO TABLE HEADER
 52 0D A2 D0 0964 2464 MOVL LNMTH\$C_PARENT(R2),R2 ;RETRIEVE PARENT TABLE HEADER ADDRESS
 0B 13 0968 2465 BEQL 3900\$;RETURN 0 BYTES IF NO PARENT TABLE
 51 09 A2 D0 096A 2466 MOVL LNMTH\$L_NAME(R2),R1 ;RETRIEVE PARENT LNMB ADDRESS
 51 11 A1 9E 096E 2467 MOVAB LNMB\$T_NAME(R1),R1 ;RETRIEVE ADDRESS AND SIZE OF
 50 81 90 0972 2468 3810\$: MOVB (R1)+,R0 ;PARENT TABLE NAME STRING

0975	2469						
0975	2470	:					
0975	2471	:	THE CURRENT ITEM IS TO BE FILLED IN WITH A VARIABLE LENGTH CHARACTER STRING				
0975	2472	:	(TABLE, STRING, OR PARENT ITEM). IF THE ITEM BUFFER PROVIDED IS NOT				
0975	2473	:	SUFFICIENTLY LARGE TO CONTAIN ALL OF THE STRING, THEN THE RETURN STATUS IS				
0975	2474	:	CHANGED TO INDICATE THIS, AND AS MUCH OF THE INFORMATION AS CAN BE STORED IS				
0975	2475	:	RETURNED.				
0975	2476	:					
0975	2477						
7E 50	7D 0975	2478	3900\$: MOVQ R0,-(SP)				:SAVE CHARACTER STRING DESCRIPTOR
7E 6B	7D 0978	2479	MOVQ (R1),-(SP)				:SAVE ITEM BUFFER DESCRIPTOR
51 6E	3C 097B	2480	MOVZWL (SP),R1				:SIZE OF ITEM BUFFER
50 04 AE	D0 097E	2481	MOVL 4(SP),R0				:ADDRESS OF ITEM BUFFER
00000000'GF	53 D4	0982	CLRL R3				:ACCESS MODE
51 8E	7D 098A	2483	JSB G^EXESPROBEW				:PROBE ITEM BUFFER
08 50	E8 098D	2484	MOVQ (SP)+,R1				:RESTORE ITEM BUFFER DESCRIPTOR
5E 08	C0 0990	2485	BLBS R0,3910\$:CONTINUE IF WRITEABLE
6E 50	D0 0993	2486	ADDL2 #8,SP				:REMOVE STRING DESCRIPTOR FROM STACK
60 11	0996	2487	MOVL R0,(SP)				:SAVE NEW STATUS
		2488	BRB 4010\$:GO RELEASE MUTEX AND RETURN
		2489					
50 6E	D0 0998	2490	3910\$: MOVL (SP),R0				:SIZE OF CHARACTER STRING TO BE RETURNED
18 13	099B	2491	BEQL 3930\$:NOTHING TO RETURN IF SIZE IS 0
51 50	B1 099D	2492	CMPW R0,R1				:ENOUGH ROOM IN BUFFER?
09 1B	09A0	2493	BLEQU 3920\$:YES - GO MOVE INFORMATION
08 AE	0601 8F	3C 09A2	MOVZWL #SSS,BUFFEROVF,8(SP)				:NO - CHANGE STATUS TO BUFFER OVERFLOW
50 51	3C 09AB	2495	MOVZWL R1,R0				:AND MOVE ONLY PART OF STRING
		2496					
62 08 BE	50 DD	09AB	2497 3920\$: PUSHL R0				:SAVE SIZE OF STRING TO BE MOVED
	50 28	09AD	MOVC3 R0,08(SP),(R2)				:MOVE THE CHARACTER STRING
	50 8ED0	09B2	POPL R0				:RESTORE NUMBER OF BYTES MOVED
		09B5					
52 5E 08 AB	08 09B5	2501	3930\$: ADDL2 #8,SP				:REMOVE STRING DESCRIPTOR FROM STACK
2C 13	D0 09B8	2502	MOVL 8(R11),R2				:WAS A RETURN LENGTH BUFFER SPECIFIED?
	09BC	2503	BEQL 3960\$:DON'T PROBE IF ONE WASN'T
62 50	B0 09C4	2504	IFNOWRT #2,(R2),3970\$:PROBE RETURN LENGTH BUFFER
21 11	09C7	2505	MOVW R0,(R2)				:MOVE IN NUMBER OF BYTES RETURNED
		2506	BRB 3960\$:GO POSITION TO NEXT ITEM
		09C9					
		2507					
		09C9					
		2508	:				
		09C9	:				
		2509	THE CURRENT ITEM IS TO BE FILLED IN WITH A LONGWORD OF INFORMATION				
		09C9	(LENGTH, MAX_INDEX, OR ATTRIBUTES ITEM).				
		09C9					
		2511	:				
		09C9					
6B 04	B1 09C9	2512	2513 3940\$: CMPW #4,(R11)				:IS ITEM BUFFER AT LEAST A LONGWORD?
50 04 AB	27 1A	09CC	BGTRU 3980\$:ERROR IF NOT
	D0 09CE	2514	MOVL 4(R11),R0				:RETRIEVE ADDRESS OF ITEM BUFFER
50 60 08 AB	52 D0	09D2	2515 IFNOWRT #4,(R0),3970\$:PROBE ITEM BUFFER
09 13	09DB	2516	MOVL R2,(R0)				:RETURN VALUE IN CURRENT ITEM
	09DF	2517	MOVL 8(R11),R0				:HAS A RETURN LENGTH BUFFER SPECIFIED?
	09E1	2518	BEQL 3960\$:DON'T RETURN LENGTH IF ONE WASN'T
60 04	B0 09E7	2519	IFNOWRT #2,(R0),3970\$:PROBE RETURN LENGTH BUFFER
	09EA	2520	MOVW #4,(R0)				:MOVE IN NUMBER OF BYTES RETURNED
	09EA	2521					
	09EA	2522					
	09EA	2523	:				
	09EA	2524	POSITION TO THE NEXT ITEM IN THE ITEM LIST.				
	09EA	2525	:				

5B 0C C0 09EA 2526
FE3F 31 09ED 2527 3960\$: ADDL2 #12 R11 ;POSITION TO NEXT ITEM
09F0 2528 BRW 3040\$;AND CONTINUE
6E 0C 3C 09F0 2530 3970\$: MOVZWL #\$\$\$ ACCVIO,(SP) ;ACCESS VIOLATION
03 11 09F3 2531 BRB 4010\$;DEALLOCATE NEW LNMB AND RETURN ERROR
6E 14 3C 09F5 2532 3980\$: MOVZWL #\$\$\$_BADPARAM,(SP) ;BAD PARAMETER SEEN
09F8 2533
09F8 2534
09F8 2535
09F8 2536 :
09F8 2537 : RELEASE THE READ LOCK ON THE LOGICAL NAME MUTEX AND RETURN.
09F8 2538 :
09F8 2539 :
54 04 AE D0 09F8 2540 4010\$: MOVL 4(SP),R4 ;ADDRESS OF PCB
F601' 30 09FC 2541 BSBW LNMSUNLOCK ;RELEASE LOCK ON LOGICAL NAME MUTEX
50 8ED0 09FF 2542 POPL R0 ;RESTORE RETURN STATUS
04 OA02 2543 RET ;RETURN
OA03 2544 .DISABLE LSB
OA03 2545
OA03 2546 : .PAGE
OA03 2547 :
OA03 2548 .END

ARMSSM_DELETE	= 00000008		LNM\$SM TERMINAL	= 00000200	
ARMSSM_EXECUTE	= 00000004		LNM\$PROBER	*****	X 02
ARMSSM_READ	= 00000001		LNM\$SEARCHLOG	*****	X 02
ARMSSM_WRITE	= 00000002		LNM\$UNLOCK	*****	X 02
CAS_MEASURE	= 00000002		LNM\$V_CASE_BLIND	= 00000019	
CNA\$CMODE	= 00000010		LNM\$V_CONFINE	= 00000001	
CNATTR	= 00000004		LNM\$V_SHAREABLE	= 00000010	
CNITMLST	= 00000014		LNM\$ CHAIN	= FFFFFFFF	
CNLOGNAM	= 0000000C		LNM\$ INDEX	= 00000001	
CNTABNAM	= 00000008		LNM\$LNMB_ADDR	= 00000009	
CTACMODE	= 00000020		LNM\$SB_ACMODE	= 0000000B	
CTATTR	= 00000004		LNM\$SB_FLAGS	= 00000010	
CTPARTAB	= 0000001C		LNM\$SB_TYPE	= 0000000A	
CTPROT	= 00000014		LNM\$BL_TABLE	= 0000000C	
CTQUOTA	= 00000010		LNM\$BM_NODELETE	= 00000010	
CTRESLEN	= 0000000C		LNM\$BM_TABLE	= 00000008	
CTRESNAM	= 00000008		LNM\$BT_NAME	= 00000011	
CTTABNAM	= 00000018		LNM\$BV_CONFINE	= 00000001	
DELETE_ACCESS	= 00000008		LNM\$BV_TABLE	= 00000003	
DNACMODE	= 0000000C		LNM\$BW_SIZE	= 00000008	
DNLOGNAM	= 00000008		LNMTHSB_FLAGS	= 00000000	
DNTABNAM	= 00000004		LNMTHSK_LENGTH	= 00000025	
DYN\$C_LNM	= 00000040		LNMTHSI_BYTES	= 00000021	
EXE\$A1OP1PROC	*****	X 02	LNMTHSI_BYTESLM	= 0000001D	
EXE\$ALOPAGED	*****	X 02	LNMTHSL_CHILD	= 00000011	
EXE\$CRELNM	00000005	RG 03	LNMTHSL_HASH	= 00000001	
EXE\$CRELNT	00000000	RG 03	LNMTHSL_NAME	= 00000009	
EXE\$DELLNM	0000000A	RG 03	LNMTHSL_ORB	= 00000005	
EXE\$LNMM_SYNTAX_DAT	*****	X 02	LNMTHSL_PARENT	= 0000000D	
EXE\$MAXACMODE	*****	X 02	LNMTHSL_QTABLE	= 00000019	
EXE\$PROBEW	*****	X 02	LNMTHSL_SIBLING	= 00000015	
EXE\$PROBEW_DSC	*****	X 02	LNMTH\$V_DIRECTORY	= 00000001	
EXE\$TRNLNM	0000000F	RG 03	LNM\$HSV_SHAREABLE	= 00000000	
EXECRELNM	00000329	R 02	LNM\$XB_FLAGS	= 00000000	
EXECRELNT	00000024	R 02	LNM\$XB_INDEX	= 00000001	
EXEDELLNM	0000066F	R 02	LNM\$SC_BACKPTR	= FFFFFF81	
EXETRNLNM	00000756	R 02	LNM\$SC_TABLE	= FFFFFF82	
HEXDIGITS	00000000	R 02	LNM\$SM_TERMINAL	= 00000002	
IPL\$ ASTDEL	= 00000002		LNM\$SM_XEND	= 00000004	
LNM\$SC_CHECK PROT	*****	X 02	LNM\$ST_XLATION	= 00000004	
LNM\$SC_TABNAMLEN	= 0000001F		LNM\$SV_XEND	= 00000002	
LNM\$DELBLK	*****	X 02	LNM\$SW_HASH	= 00000002	
LNM\$DELETE_LNMB	*****	X 02	LNM\$HEX_TAB_LEN	= 00000014	
LNM\$DELETE_TAB	*****	X 02	ORB\$C_LENGTH	= 00000058	
LNM\$FIRSTTAB	*****	X 02	ORB\$K_LENGTH	= 00000058	
LNM\$INIT PROT	*****	X 02	PCBSL_EPID	= 00000064	
LNM\$INSLGTTAB	*****	X 02	PCBSQ_PRIV	= 00000084	
LNM\$LOCKR	*****	X 02	PMS\$GE_LOGNAM	*****	X 02
LNM\$LOCKW	*****	X 02	PR\$ IPC	= 00000012	
LNM\$SM_CASE_BLIND	= 02000000		PRV\$V_SYSNAME	= 00000002	
LNM\$SM_CONCEALED	= 00000100		PSL\$C_USER	= 00000003	
LNM\$SM_CONFINE	= 00000002		PSL\$M_PRVMOD	= 00C00000	
LNM\$SM_CREATE_IF	= 01000000		QUOTA_ACCESS	= 00000004	
LNM\$SM_CRELOG	= 00000004		READ_ACCESS	= 00000001	
LNM\$SM_EXISTS	= 00000400		SSS\$ACCVIO	= 0000000C	
LNM\$SM_NO_ALIAS	= 00000001		SSS\$BADPARAM	= 00000014	
LNM\$SM_SHAREABLE	= 00010000		SSS\$BUFFEROVF	= 00000601	

SYS
Pse

PSE

\$\$\$
Pha

Ini
Com
Pas
Sym
Pas
Sym
Pse
Crc
Ass
The
257
The
362
2 p
Mac

-S2
-S2
TO1
58
The
MAC

```
SS$_EXLNMQUOTA      = 0000224C
SS$_INSMEM          = 00000124
SS$_IVLOGNAM        = 00000154
SS$_IVLOGTAB        = 0000015C
SS$_NOLOGNAM        = 000001BC
SS$_NORMAL           = 00000001
SS$_RESULTOVF        = 00000214
SS$_SUPERSEDE        = 00000631
TRACMODE            = 00000010
TRATTR               = 00000004
TRITMLST             = 00000014
TRLOGNAM             = 0000000C
TRTABNAM             = 00000008
WRITE_ACCESS         = 00000002
```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YF\$SLNM	00000A03 (2563.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
Y\$EXEPAGED	00000014 (20.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.07	00:00:00.34
Command processing	106	00:00:00.55	00:00:01.48
Pass 1	403	00:00:15.02	00:00:31.49
Symbol table sort	0	00:00:01.91	00:00:03.39
Pass 2	417	00:00:05.92	00:00:13.20
Symbol table output	1	00:00:00.10	00:00:00.22
Psect synopsis output	0	00:00:00.03	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	959	00:00:23.60	00:00:50.15

The working set limit was 2100 pages.

94360 bytes (185 pages) of virtual memory were used to buffer the intermediate code.

There were 70 pages of symbol table space allocated to hold 1116 non-local and 159 local symbols.

2548 source lines were read in Pass 1, producing 32 object records in Pass 2.

25 pages of virtual memory were used to define 24 macros.

! Macro library statistics !

Macro library name

-\$255\$DUA2B:[SYS.OBJ]LIB.MLB;1
-\$255\$DUA2B:[SYSLIB]STARLET.MLB;2
TOTALS (all libraries)

Macros defined

11
10
21

1208 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:SYSLNM/OBJ=OBJ\$:SYSLNM MSRC\$:SYSLNM/UPDATE=(ENH\$:SYSLNM)+EXECML\$/LIB

0386 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSARAM
LIS

SYSMTRACE
LIS

SYSIMGSTA
LIS

SYSLHM
LIS

SYSLOGNAME
LIS

SYSLKSET
LIS

SYSMAILBX
LIS